Requirements Prioritisation and Retrospective Analysis for Release Planning Process Improvement

Lena Karlsson

LUND UNIVERSITY

Department of Communication Systems
Lund Institute of Technology

ISSN 1101-3931
ISRN LUTEDX/TETS--1081--SE+192P

Printed in Lund, Sweden by E-kop

September 2006

To mum and dad

This thesis is submitted to the Research Education Board of the Faculty of Engineering at Lund University, in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering.

Contact Information:

Lena Karlsson
Department of Communication Systems
Lund University
P.O. Box 118
SE-221 00 LUND
Sweden

Tel: +46 46 222 03 65
Fax: +46 46 14 58 23
E-mail: lena.karlsson@telecom.lth.se

Abstract

The quality of a product can be defined by its ability to satisfy the needs and expectations of its customers. Achieving quality is especially difficult in market-driven situations since the product is released on an open market with numerous potential customers and users with various wishes. The quality of the software product is to a large extent determined by the quality of the requirements engineering (RE) and release planning decisions regarding which requirements that are selected for a product. The goal of this thesis is to enhance software product quality and increase the competitive edge of software organisations by improving release planning decision-making.

The thesis is based on empirical research, including both qualitative and quantitative research approaches. The research contains a qualitative survey of RE challenges in market-driven organisations based on interviews with practitioners. The survey provided increased understanding of RE challenges in the software industry and gave input to the continued research. Among the challenging issues, one was selected for further investigation due to its high relevance to the practitioners: requirements prioritisation and release planning decision-making. Requirements prioritisation techniques were evaluated through experiments, suggesting that ordinal scale techniques based on grouping and ranking may be valuable to practitioners. Finally, a retrospective method called PARSEQ (Post-release Analysis of Requirements SElection Quality) is introduced and tested in three case studies. The method aims at evaluating prior releases and finding improvement proposals for release planning decision-making in future release projects. The method was found valuable by all participants and relevant improvement proposals were discovered in all cases.

Acknowledgements

This work was funded by Lund University Faculty of Engineering as well as the Swedish Agency for Innovation Systems (VINNOVA) under the grant for the Center for Applied Software Research at Lund University (LUCAS) and the ITEA project for Embedded Systems Engineering in Collaboration (MERLIN).

First and foremost, I would like to thank my supervisor Björn Regnell for sharing his great creativity and knowledge. Thanks also to Martin Höst, Per Runeson, and Thomas Thelin for support and assistance. Many thanks to all past and present colleagues, co-authors, and friends at the Department of Communication Systems, at Blekinge Institute of Technology, at University of Skövde, and in the MERLIN project. Thanks to all anonymous individuals who have contributed to the thesis: interviewees and focus group participants, experiment subjects, industrial case study participants, and reviewers. A special thanks to Per Klingnäs and Mikael Jönsson for excellent development of the PARSEQ tool. Finally, thanks to my family for constant encouragement, and to friends who have stood by me through rain and sunshine.

Lena Karlsson
September, 2006

Contents

Introduction 3
1. Research Focus 9
2. Related Work 10
3. Research Approach. 17
4. Research Results. 25
Part I 33
1. Introduction 34
2. Related Work 35
3. Research Methodology 38
4. Results 45
5. Discussion. 59
6. Conclusions. 64
Appendix 66
Part II 69
1. Introduction 70
2. Related Work 72
3. Pair-wise Comparisons versus Planning Game Partitioning - Experiments on Requirements Prioritisation Techniques 81
4. Evaluating the Practical Use of Different Measurement Scales in Requirements Prioritisation 109
5. Closing Remarks 124
Appendix 126
Part III. 129
1. Introduction 130
2. Related Work 132

3. The PARSEQ Method *136*
4. The PARSEQ Tool-Support. *141*
5. The PARSEQ Method in Case Studies *144*
6. Discussion *176*
7. Conclusions *181*
References 185

Introduction

Software continually becomes a more and more important part of an increasing number of products. Several different domains need to deal with software development, e.g. the automotive industry, developers of medical IT, and developers of commercial products such as mobile phones and digital cameras. The intangible and flexible nature of software causes software projects to be over-represented in project failure statistics. Typical problems include lack of functionality, poor quality, budget overruns and missed deadlines (The Standish Group, 2001).

Quality can be defined as *the degree to which a system, component, or process meets customer or user needs or expectations* (IEEE, 1990). Even if a product is delivered on time and within budget, it may be a failure due to poor quality if it does not meet customer and user expectations. In particular, *market-driven* organisations, which release their products on an open market with numerous potential customers and users, experience challenges with quality. Satisfying customer expectations is very difficult when the customers are diverse and have different opinions.

In software product development the customer expectations are elicited, analysed, specified, and validated in an activity called requirements engineering (RE) (Sommerville, 2001). The activity lays the foundation for successful planning and development of the product before release to the market. In a competitive environment, such as the one experienced by market-driven organisations, it is essential to plan product releases with time-to-market in mind. Release planning is where requirements engineering for market-driven software product development meets the market perspective. Selecting a subset of requirements for realisation in a certain release is as complex as it is important for the success of the product (Carlshamre, 2002).

The main goal of the research presented in this thesis is to enhance software product quality and to increase the competitive edge of software organisations. By developing and applying methods and techniques for assessing and improving RE and release planning, the software product quality is expected to improve. The main contributions are: increased understanding of RE challenges in the software industry based on a qualitative survey, evaluation of requirements prioritisation techniques based on experiments, and a method for retrospective analysis of release planning decision-making evaluated in case studies.

The thesis starts with an introduction to the research area and the research focus. Following the introduction there are three parts, each including between one and three papers. In order to avoid repetition and redundancy, each part has one introduction and one concluding section. That is, when two or three papers are combined, the introduction and conclusion sections have been integrated. In total, six papers are included partly or completely in the thesis.

- **Introduction.** This section describes the background of the research. Section 1 presents the research focus and research questions examined in the thesis. Section 2 continues with a description of related work to put the research into context. The research approach and validation issues are described in Section 3, before the research results and contributions are presented and future research is discussed in Section 4.

- **Part I: Requirements Engineering Challenges in Industry.** The first part presents a qualitative survey of RE challenges in market-driven organisations (Paper 1). The paper describes challenging areas within RE experienced by 14 interviewed practitioners. Among the many challenges we find issues related to release planning and requirements prioritisation.

- **Part II: Evaluation of Requirements Prioritisation Techniques.** The work in Part II is focused on evaluating different techniques for requirements prioritisation since it is an important activity in release planning. The second part includes two studies: the first one describes two experiments comparing different requirements prioritisation techniques (Paper 2) and the second one presents an archive analysis examining results from prioritisation sessions (Paper 3).

- **Part III: Retrospective Analysis for Release Planning Decisions.** Release planning process improvement is investigated in the third part. A method for retrospective analysis of release planning decision-making is presented, as well as results from three industrial case studies (Paper 4 and 5). In addition, we present tool support which was used and evaluated in one of the case studies (Paper 6).

List of Papers

The thesis is based on the following six papers:

1. REQUIREMENTS ENGINEERING CHALLENGES IN MARKET-DRIVEN SOFTWARE DEVELOPMENT – AN INTERVIEW STUDY WITH PRACTITIONERS
Lena Karlsson, Åsa G. Dahlstedt, Björn Regnell, Johan Natt och Dag, Anne Persson
Accepted for publication in Information and Software Technology: Special Issue on Understanding the Social Side of Software Engineering, Qualitative Software Engineering Research, 2007.

2. PAIR-WISE COMPARISONS VERSUS PLANNING GAME PARTITIONING - EXPERIMENTS ON REQUIREMENTS PRIORITISATION TECHNIQUES
Lena Karlsson, Thomas Thelin, Björn Regnell, Patrik Berander, Claes Wohlin
Accepted for publication in Empirical Software Engineering Journal, 2006.

3. EVALUATING THE PRACTICAL USE OF DIFFERENT MEASUREMENT SCALES IN REQUIREMENTS PRIORITISATION
Lena Karlsson, Martin Höst, Björn Regnell
Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE'06), Rio de Janeiro, Brazil, September 2006.

4. CASE STUDIES IN PROCESS IMPROVEMENT THROUGH RETROSPECTIVE ANALYSIS OF RELEASE PLANNING DECISIONS
Lena Karlsson, Björn Regnell, Thomas Thelin
Accepted for publication in International Journal of Software Engineering and Knowledge Engineering: Special Issue on Requirements Engineering Decision Support, December 2006.

5. RETROSPECTIVE ANALYSIS OF RELEASE PLANNING DECISIONS IN A PRODUCT LINE ENVIRONMENT - A CASE STUDY
Lena Karlsson, Björn Regnell

Submitted, 2006.

6. INTRODUCING TOOL SUPPORT FOR RETROSPECTIVE ANALYSIS OF RELEASE PLANNING DECISIONS
Lena Karlsson, Björn Regnell
Proceedings of the 7th International Conference on Product Focused Software Process Improvement (PROFES'06), Amsterdam, the Netherlands, June 2006, pp. 19-33.

Related Publications

The following publications are related but not included in the thesis:

7. UNDERSTANDING SOFTWARE PROCESSES THROUGH SYSTEM DYNAMICS SIMULATION: A CASE STUDY
Carina Andersson, Lena Karlsson, Josef Nedstam, Martin Höst, Bertil I Nilsson
Proceedings of the 9th IEEE Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden, April 2002, pp. 41-48.
(This paper presents a simulation model which worked as foundation for the model presented in Paper 10.)

8. CHALLENGES IN MARKET-DRIVEN REQUIREMENTS ENGINEERING - AN INDUSTRIAL INTERVIEW STUDY
Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, Anne Persson
Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, September 2002, pp. 37-49.
(This paper presents intermediate results from Paper 1 and is based on the first seven interviews.)

9. POST-RELEASE ANALYSIS OF REQUIREMENTS SELECTION QUALITY - AN INDUSTRIAL CASE STUDY
Lena Karlsson, Björn Regnell, Joachim Karlsson, Stefan Olsson
Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Velden, Austria, June 2003, pp. 47-56.
(This paper presents the first of the case studies described in Paper 4.)

10. AN ANALYTICAL MODEL FOR REQUIREMENTS SELECTION QUALITY EVALUATION IN PRODUCT SOFTWARE DEVELOPMENT
Björn Regnell, Lena Karlsson, Martin Höst
Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE'03), Monterey Bay, California, the USA, September 2003, pp. 254-263.
(This paper is summarised in the Introduction, Section 2.3. It presents results from a survey, which motivates the evaluation and improvement of requirements selection quality.)

11. MARKET-DRIVEN REQUIREMENTS ENGINEERING PROCESSES FOR SOFTWARE PRODUCTS - A REPORT ON CURRENT PRACTICES
Åsa G. Dahlstedt, Lena Karlsson, Johan Natt och Dag, Björn Regnell, Anne Persson
1st International Workshop on COTS and Product Software (RECOTS'03), Monterey Bay, California, USA, September 2003.
(This paper presents intermediate results from Paper 1 and is based on the first seven interviews. The paper compares the discovered challenges to the characteristics of market-driven software development reported in literature.)

12. IMPROVING REQUIREMENTS SELECTION QUALITY IN MARKET-DRIVEN SOFTWARE DEVELOPMENT
Lena Karlsson
Licentiate thesis, ISRN LUTEDX/TETS-1063-SE+132P, Dept. of Communication Systems, Lund University, Sweden.
(The licentiate thesis includes Papers 7, 8, 9, 10, and an early version of Paper 13.)

13. REQUIREMENTS PRIORITISATION: AN EXPERIMENT ON EXHAUSTIVE PAIR-WISE COMPARISONS VERSUS PLANNING GAME PARTITIONING
Lena Karlsson, Patrik Berander, Björn Regnell, Claes Wohlin
Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE'04), Edinburgh, UK, May 2004, pp. 145-154.
(This paper presents the first of the two experiments in Paper 2.)

14. INVESTIGATION OF REQUIREMENTS SELECTION QUALITY IN MARKET-DRIVEN SOFTWARE PROCESSES USING AN OPEN SOURCE DISCRETE EVENT SIMULATION FRAMEWORK
Björn Regnell, Bengt Ljungquist, Thomas Thelin, Lena Karlsson
Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim'04), Edinburgh, UK, May 2004, pp. 84-93.
(This paper introduces a simulation framework for the analytical model of requirements selection quality presented in Paper 10.)

15. ALIGNING THE REQUIREMENTS ENGINEERING PROCESS WITH THE MATURITY OF MARKETS AND PRODUCTS
Lena Karlsson, Björn Regnell
Proceedings of the 10th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'04), Riga, Latvia, June 2004, pp. 69-74.
(This paper describes market-driven RE from a lifecycle perspective, since it was discussed by interviewees in Paper 8.)

16. HOW EVALUATION TECHNIQUES INFLUENCE THE RE-TOOL EVALUATION: AN EXPERIMENT.
Raimundas Matulevicius, Lena Karlsson, Guttorm Sindre
Proceedings of the European Software Process Improvement Conference (EuroSPI'04), Trondheim, Norway, November 2004, pp. I3B11-I3B16.
(This paper investigates RE tools, which is related to the topic in Paper 6.)

17. Comparing Ordinal and Ratio Scale Data in Requirements Prioritisation
Lena Karlsson, Björn Regnell
Proceedings of the 3rd International Workshop on Comparative Evaluation in Requirements Engineering (CERE'05), Paris, France, August 2005, pp 21-29.
(This paper presents the measures for comparing prioritisation results from different measurement scales, which are further evaluated with a larger data set in Paper 3.)

18. A Case Study in Retrospective Analysis of Release Planning in an Agile Project
Lena Karlsson, Björn Regnell, Thomas Thelin
1st Workshop on the Interplay of Requirements Engineering and Project Management in Software Projects (REProMan'05), Paris, France, August 2005.
(This paper presents the second of the case studies presented in Paper 4.)

Contribution Statement

The author of the thesis is the main author of the six included papers. This means responsibility for running the research process, dividing the work between coauthors and conducting most of the writing. Paper 1 and 2 were produced in cooperation with other universities and have five authors each. In both cases, a lot of the design and analysis was performed together with coauthors, while most of the writing and division of work was performed by the main author. The research in Paper 3, 4, and 5 was performed primarily by the main author, who designed and conducted most of the work, as well as reported on the studies. Paper 6 describes a tool which was designed by the author, but developed by two external developers. The paper is written primarily by the main author.

1. Research Focus

The goal of this research is to find means for improving the RE and release planning processes. The purpose is to enhance software product quality and increase the competitive edge of software organisations. The main research questions that have been investigated are:

RQ1. Which challenges related to RE are experienced by practitioners in the market-driven software development industry?

RQ2. How can requirements prioritisation techniques be characterised and compared?

RQ3. How can retrospective analysis be used to evaluate and improve the release planning process?

The research questions RQ1-RQ3 correspond to Part I-III in the thesis. The relation between the research questions is illustrated in Figure 1.

RQ1 was posed in order to discover and understand RE challenges experienced by practitioners and was used to select focus for the research. Among the many challenges that appeared in the qualitative survey, issues regarding requirements prioritisation and release planning emerged. Therefore, these areas were targeted in the continued research.

RQ2 was examined to understand and compare requirements prioritisation techniques since requirements prioritisation is a vital activity for release planning. The characteristics of different techniques were identified in experiments and their potential support for release planning decision-making was examined.

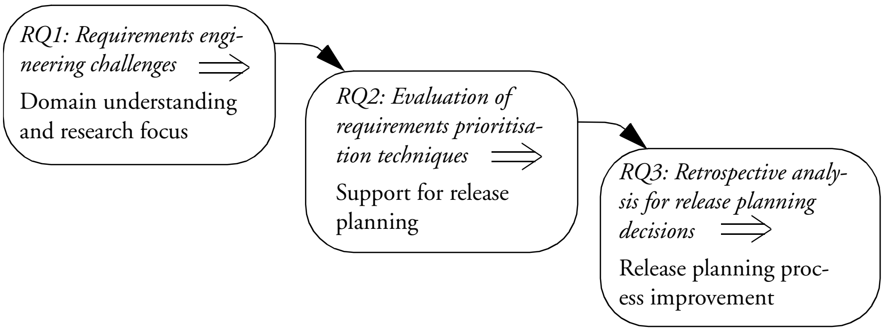

RQ1: Requirements engineering challenges ⟹
Domain understanding and research focus
RQ2: Evaluation of requirements prioritisation techniques ⟹
Support for release planning
RQ3: Retrospective analysis for release planning decisions ⟹
Release planning process improvement

Figure 1. *The three parts of the thesis*

RQ3 aims at improving the release planning process through retrospective analysis. The results from RQ2 were used to create a method for evaluating the release planning process and discovering possible improvements. The retrospective analysis method was applied in three industrial case studies with different characteristics in order to investigate its possibilities and limitations.

2. Related Work

This section provides some theoretical background to the requirements engineering area and describes the context of the research in the thesis. The successive subsections describe related work in general, while in Section 4 related work is discussed in relation to the thesis findings.

Software engineering is an engineering discipline whose goal is to cost-effectively develop software systems. This includes all aspects of software development; from the early stages of system specification through to maintaining the system after it is put into use (Sommerville, 2001). *Software requirements* are by the Software Engineering Body of Knowledge (SWEBOK, 2004) defined as properties that must be exhibited in order to solve some problem of the real world. In other words, requirements define what the system should do, i.e. what functionality and qualities the system shall include. Thus, *requirements engineering* regards the process of identifying, analysing, documenting, validating and managing these software properties (Lauesen, 2002).

Systems engineering is concerned with all aspects of computer-based systems development, including hardware, software, electrical and mechanical engineering, thus software engineering is part of this area. The software in these systems is *embedded* in a hardware system and must respond, in real-time, to events from the system's environment (Sommerville, 2001). In the thesis, the term *product* is also used to refer to a system that partly or completely consists of software.

2.1 Requirements Engineering

Traditionally, RE takes place in the beginning of every project, and results in a specification that defines the product to be developed. This view is based on the *Waterfall model* (Royce, 1970), where requirements engineering is followed by design, implementation, testing and maintenance activities. However, this cascade process may not be the most appropriate in practice, since the flexible nature of software requires the development process to be more iterative and evolutionary. New and changed requirements appearing during development calls for continuous RE efforts.

The four main activities in the RE process, as defined by Sommerville (2001), are illustrated in Figure 2. The feasibility study is performed before starting with elicitation, and the activities are, in practice, performed iteratively in order to handle changing requirements. In addition, requirements management is performed continually throughout the product life-cycle to understand and control requirements changes.

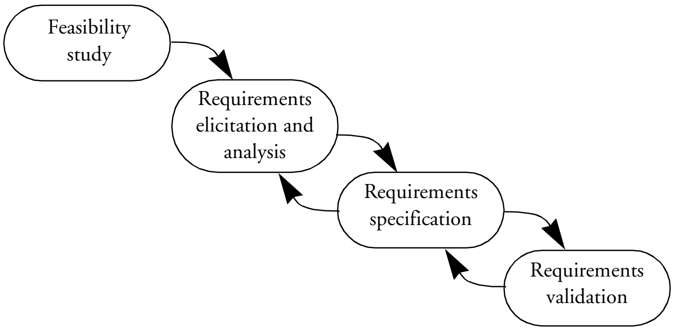

Feasibility study
Requirements elicitation and analysis
Requirements specification
Requirements validation

Figure 2. *The requirements engineering process*

- *Feasibility study* is performed to decide whether or not it is worth carrying on with development. The system should contribute to the overall objectives of the organisation and be possible to implement with the current technology and within the given cost and schedule constrains.

- *Requirements elicitation and analysis* starts with gaining application domain understanding and moves on to collecting requirements from stakeholders for the system. Next, the requirements are classified and conflicting views among stakeholders are resolved. In any set of requirements, some are more important than others. Prioritisation is performed to discover the most important requirements. Finally, the requirements are checked for completeness, consistency, and accordance with the stakeholders' wishes.

- *Requirements specification* involves documenting the elicited functional and non-functional requirements in detail. Non-functional requirements are also called quality requirements and affect how well a system must perform its functions (Lauesen, 2002). In addition, the specification may include purpose and scope of the product, user characteristics, and development constraints.

- *Requirements validation* involves showing that the requirements actually define the system which the customer wants and is concerned with finding problems with the requirements. Validation can be performed with different techniques such as reviews or prototyping.

2.2 Market-Driven Requirements Engineering

Products can be divided into different categories depending on the type of market where the product is vended. Among the early work that characterises the differences between *customer-specific* and *market-driven* development is the field study by Lubars et al. (1993). The authors investigated differences between the two types of development in the areas of requirements definition, specification and validation.

In the customer-specific case (also called *bespoke* or *contract-driven*) the product is ordered by a specific customer and the supplier develops and maintains the product for that customer. The customer often represents a large organisation such as a military, governmental or financial institution. A product contract is negotiated with the customer, describing what the product shall include, when it shall be delivered, and how much it will cost.

Market-driven software systems or products (also called *packaged* or *commercial-of-the-shelf*) are developed for an open market. The customer may be another organisation or a consumer and the products may be, for example, computer packages, development tools, or mobile phones. In both cases there is a large range of *potential* customers on a mass market and suppliers need to take diverse needs into account.

The characteristics of market-driven development have been described by Lubars et al. (1993), Potts (1995), Sawyer (2000), and Carlshamre (2001). Some of their findings are summarised below.

The characteristics of market-driven RE include, for example, that the mass market product often has a life cycle with several consecutive releases and it lasts as long as there is a market for it. Therefore, release planning is an important activity. A highly important issue is to have shorter time-to-market than the competitors, in order to yield high market shares and be successful on the market.

Requirements are often invented by developers or elicited from a set of potential customers since there is no single customer to ask. This may yield too many requirements with respect to the available resources for one release. It is necessary to make estimations of implementation effort and market value in order to prioritise and select a set that will fit the market and corporate strategy. Requirements are prioritised within the market-driven developing organisation before release planning, while in the customer-specific situation the requirements are negotiated and contracted with the customer.

Many organisations do in fact deal with both market-driven and customer-specific projects. In this thesis, we focus mainly on the market-driven aspects of these organisations.

2.3 Release Planning Decision-Making

Release planning is one of the specific RE activities conducted in market-driven organisations, along with prioritisation and cost estimation. Release planning can be described as selecting an optimal subset of requirements for realisation in a certain release. Thus, it is a major determinant of the success of the software product (Carlshamre, 2002).

Wohlin and Aurum (2005) identified relevant criteria for release planning based on a survey with practitioners. One of the criteria regarded as relevant to all respondents in the survey was the actual cost-benefit trade-off for implementing a requirement. This is similar to the criteria used by Karlsson and Ryan (1997) in the cost-value approach. The approach is based on optimising the relation between the requirements' value and cost in order to achieve stakeholder satisfaction.

The requirements selection and release planning process is supported by requirements prioritisation, which can be defined as the activity during which the most important requirements for a system are identified (Sommerville, 2001). Issues that determine the priority of a requirement include importance to users and customers, implementation costs, logical implementation order, and financial benefit (Lehtola et al., 2004). There

Table 1. Summary of prioritisation techniques

Technique	Description	References
Planning game	Grouping and ranking requirements on an ordinal scale. Usually based on the criteria value, cost and risk.	Beck, 2005
Pair-wise comparisons	Comparisons between all pairs of requirements. Based on the Analytical Hierarchy Process (AHP) and the result is on a ratio scale. Usually based on cost and value criteria.	Saaty, 1980; Karlsson, 1997
Numeral Assignment	Grouping requirements in e.g. 3 or 5 groups, usually based on customer value. The result is presented on an ordinal scale.	IEEE, 1998; Karlsson, 1996
$100 test	Also called cumulative voting and is suitable in distributed environments. Based on assigning fictional money to requirements and the result is on a ratio scale.	Leffingwell and Widrig, 2000
Wiegers' method	Combines the customer value, penalty if the requirement is not implemented, implementation cost, and risks. The result is on a ratio scale.	Wiegers, 1999

are several prioritisation techniques described in literature, but prioritisation practice is informal in many companies (Lehtola and Kauppinen, 2006). Some of the prioritisation techniques are summarised in Table 1. For more details regarding requirements prioritisation techniques, see e.g. Moisiadis (2002) and Berander and Andrews (2005), as well as Part II of this thesis.

The selection of requirements for a release is often made in several steps of the RE process, since release plans are revised and changed throughout development as more knowledge is gained about market expectations and development progress. Starting out with a large set of potential requirements, the selection brings the number down for each activity in the requirements process. This is illustrated in Figure 3, which is adapted from Regnell et al. (2003) (Paper 10). The discarded requirements are typically stored in a database for investigation in future releases. During *elicitation*, decisions concern which stakeholder representatives to consult to elicit ideas for new features. Then there is often a *screening* activity performed to make a first quick assessment to decide whether a requirement is worth spending more time on. Requirements that are clearly out of scope for the next release are rejected

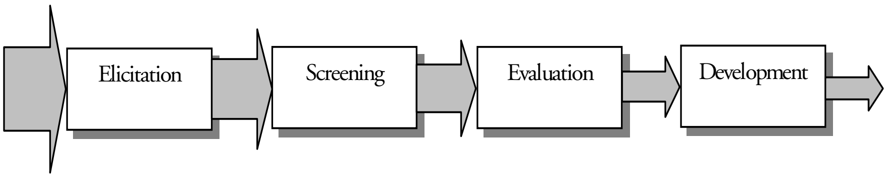


Figure 3. *The RE process from a requirements selection viewpoint*

in order to avoid overload in the process (Regnell et al., 1998). The *evaluation* activity includes prioritising requirements and identifying requirements that are interdependent. Finally, the requirements can be re-assessed during *development* and decisions regarding postponing or removing requirements can be made based on the information gained during implementation.

In Regnell et al. (2003), the requirements selection process is described as a queuing network model, with parameters for arrival rates, service rates, number of servers, and probability for a requirement to remain in the process. The parameters were estimated based on a survey with practitioners from companies developing software-intensive products. The survey indicates that, on average, only 21% of all incoming requirements are good enough to be implemented with regard to market opportunities, product strategy and development resources. Evidently it is difficult to determine which of the incoming requirements to select for implementation. Furthermore, a majority of the respondents estimate that only 25-50% of the requirements selection decisions made in their organisation are correct. It appears that there is a large opportunity for improving the release planning process and thereby the requirements selection quality.

2.4 Retrospective Analysis for Process Improvement

Several different approaches to Software Process Improvement (SPI) have been suggested. Among the common ones are maturity models such as the Capability Maturity Model (CMM) (Paulk et al., 1993) and standards such as ISO9000 (www.iso.org). These approaches aim at assessing and improving the process through a set of principles or practices. However, many of those who have applied CMM have been disappointed about the time and costs required for the assessment and improvement (Herbsleb and Goldenson, 1996).

An alternative to using pre-defined principles and practices is to base the process improvement effort on the experiences of your own organisation or project. One such approach is the Experience Factory (Basili et al., 1994). It is aimed at capitalising and reusing lifecycle experiences and products through processing project information and data, and giving feedback to project activities. Basili et al. (1994) state that reuse of knowledge is the basis of improvement. Another approach based on internal experiences is the *retrospective analysis*, acknowledged as one of the most important steps toward improving the software process (Kerth, 2001). Retrospective analysis has been used under different names such as *postmortem analysis* (Birk et al., 2002), *postmortem project evaluation* (Ulrich and Eppinger, 2000), and *postmortem review* (Dingsoyr, 2005). Retrospective analysis is usually performed after the project is finished and may consist of an open-ended discussion of the strengths and weaknesses of the project plan and execution. Sometimes it is facilitated by an outside consultant or someone with an objective view of the project. At the end of the session, a postmortem report is prepared as a formal closing of the project, which is then used in the project planning stage of future projects (Ulrich and Eppinger, 2000).

Retrospective analysis is also recognised as a valuable method for *knowledge management,* which promotes an organisation's intellectual capital (Rus and Lindvall, 2002). It focuses on the individual as an expert and bearer of important knowledge that he or she can systematically share within the organisation. The retrospective analysis has also been used in the agile community, which uses the concept *process refactoring* (Collins and Miller, 2001) in order to emphasise the continuous approach to process improvement. Instead of waiting until a project is finished, process refactoring takes place *during* the project in order to improve before it is too late. Retrospective analysis is discussed further in Part III of the thesis.

Some of the more recent work focus on the outcome of the retrospective analysis, i.e. how the findings are reported and used. Dingsoyr (2005) describes a case study at a company where every project wrote an experience report, but these were seldom read by other projects. It seems important to consider how the results from the retrospective analysis are reported and transferred to future projects. One way to make postmortem analysis results more accessible is presented by Schalken et al. (2006). The authors present a method to derive findings from a set of postmortem review reports and transform the qualitative information into quantitative information. The results may provide guidance in a process improvement initiative. Desouza et al. (2005) describe an evaluation of two different outcomes from the retrospective process: traditional reports and stories. The differences in structure, cost of preparation, richness of knowledge, and ease of comprehension require each project to decide which one is more appropriate.

3. Research Approach

This section gives an overview of the different research approaches that are used in the thesis. It also describes certain validity issues that need to be considered for each approach. The section starts with an introduction of some research methodology concepts and continues with three sections describing the research approach and validity for each research question. The validity is discussed from a research design perspective, i.e. which measures that were taken during design to increase validity of the results. In Section 4, the validity is discussed from the perspective of the results, e.g. limitations of the results and the results in relation to literature.

One main research approach is used for each research question. The first research question is answered by a qualitative survey, the second by experiments, and the third by a series of case studies.

3.1 Research Designs

There are two main types of research designs: *fixed* and *flexible* designs (Robson, 2002). The fixed design, also called *quantitative*, deals with designs that are highly pre-specified and prepared. A conceptual framework or theory is required in order to know in advance what to look for. It is often concerned with quantifying a relationship or comparing two or more groups and the results are often *prescriptive*, i.e. it suggests a solution, method or tool that is more appropriate than another.

The flexible design, also called *qualitative*, relies on qualitative data and requires less pre-specification. The design is intended to evolve and develop during the research process as the researchers gain more knowledge. The flexible design is concerned with studying objects in their natural setting and is often *descriptive*, i.e. it describes some issue of the real world. Qualitative data may include numbers, but are to a large extent focused on words. Many times, however, a design may include both fixed and flexible methods and yield both quantitative and qualitative data.

The research in this thesis uses three major types of methods: *survey*, *experiment* and *case study*. Surveys and case studies can be both fixed and flexible, while experiments are typically fixed (Wohlin et al., 2000). Pure qualitative research designs include strategies such as *grounded theory* and *ethnography*, which are discussed briefly in Part I of this thesis and more in depth by Robson (2002).

All research designs and approaches have certain validity issues that need to be considered. In fixed designs there are mainly four types of validity: conclusion, internal, construct and external validity. In flexible designs there is a different set of validity issues, which regards description, interpretation and theory (Robson, 2002). In addition to these, there are matters of respondent and researcher bias, i.e. when people involved in the research, deliberately or not, affect the results.

3.2 Survey on RE Challenges

This section describes the qualitative survey methodology, which was used to answer the first research question. It also discusses validity issues for flexible designs and for the performed survey.

SURVEYS. Surveys can be both flexible and fixed, i.e. include different degrees of pre-specification. Survey is a wide term that includes everything from open-ended interviews (typically flexible) to questionnaires with closed questions (typically fixed). While questionnaires can reach a large set of people and provide data that is easy to analyse, there is a risk of low response rates and questionnaires can be prone to misunderstandings. Interviews have higher response rates and the interviewer may explain and clarify questions during the session to avoid misunderstandings. However, there are disadvantages such as high time consumption and that the interviewer may impose a bias (Robson, 2002). The purpose of surveys is to understand, describe, explain or explore the population (Wohlin et al., 2000).

VALIDITY IN FLEXIBLE DESIGNS. The three main validity issues in flexible designs are presented here. A more detailed presentation is available in Robson (2002).

1. *Description* is regarding the accuracy and completeness of the data. Thus, if interviews occur they should be audio-taped and possibly transcribed in order to keep all data for reference and analysis.

2. *Interpretation* implies that frameworks and theories shall emerge from the knowledge gained during the research, instead of being predetermined and biased. This is achieved by analysing and demonstrating how the interpretation was reached.

3. *Theory* is closely related to interpretation, but regards the threat of not considering alternative explanations of the phenomena under study. This is confronted by continuously revising and refining the theory until it accounts for all known cases.

RESEARCH QUESTION 1.
Which challenges related to RE are experienced by practitioners in the market-driven software development industry?

The first research question was investigated by conducting a flexible survey at eight different Swedish companies involved in market-driven software development. Fourteen practitioners participated in interviews, which were recorded and later transcribed and analysed with support from a commercial data analysis tool. The interviews were open-ended and had a flexible structure, and took different shape depending on the responses. To validate intermediate results, a focus group meeting was held with five RE practitioners.

The data was stored both on audio tape and as printed transcriptions in order to keep data complete and accurate. Thus, the *description validity* is taken into consideration. In most interviews, two or three researchers participated and extensive notes were taken. During analysis, three researchers with different research interests examined the data and drew different conclusions that were later discussed. Such discussions from different angles help to ensure that conclusions were not emerged through prejudices, but through the knowledge gained during the study. Thereby, we believe that the *interpretation validity* is regarded.

The eight companies were of different size and age, and from different business areas, and at six of the companies, interviews were held with two people in different organisational positions. In that manner, we believe that the gained knowledge is based on many different aspects and the results reflect a broad image of the reality. The selection of companies from different categories of age, size and business were made with the intent to regard *theory validity.*

Since the study is based on a flexible design we do not intend to generalise the results to a larger population, but only to the setting under study. The intention with the survey was to gain understanding of RE and describe RE challenges. Based on the gained understanding we could find areas in need of further research.

3.3 Experiments on Prioritisation Techniques

This section describes the methodology of experiments as it was used to answer the second research question. Although Paper 3 is not a controlled experiment but an archive analysis, the methodology and validity issues of experiments can be applied. It also discusses validity in fixed designs and presents how the validity threats were handled in the studies.

EXPERIMENTS. Experiments are used when we want control over the situation and wish to manipulate the behaviour. Results are often reported as averages and proportions, thus it is a quantitative design. Controlled experiments involve more than one treatment to compare the outcomes and enable statistical analysis. As other fixed designs, the experiment is theory-driven and requires a substantial amount of conceptual understanding from the start (Robson, 2002).

The design of an experiment should be made so that the objects involved represent all the methods or tools we are interested in. The strength of an experiment is that we can investigate in which situation the claims are true and they provide a context in which certain methods or tools are recommended for use (Wohlin et al., 2000).

VALIDITY IN FIXED DESIGNS. Some of the validity problems encountered in fixed designs are briefly described here, while more thorough presentations are available in Robson (2002) and Wohlin et al. (2000).

1. *Conclusion validity* is concerned with the relationship between the treatment and the outcome. We want to make sure that there is a statistical relationship, i.e. with a given significance. The threats are concerned with choice of statistical tests, sample sizes and care taken in the implementation and measurement of the experiment.

2. *Internal validity* is needed to make sure that the relationship between the treatment and outcome is a causal relationship, i.e. that the treatment actually caused the result. Threats to internal validity concern issues such as how the subjects are selected and divided into classes, and how subjects are treated during the experiment, i.e. factors that can make the experiment show behaviour that is not due to the treatment.

3. *Construct validity* is concerned with the relation between the theory and the observation and refers to the extent to which the experiment setting actually reflects the construct under study. Using multiple strategies to measure the same thing may improve the construct validity and ensure that the result is an effect of the treatment.

4. *External validity* regards generalisability of the setting and the subjects. Although internal validity is regarded, i.e. there is a causal relationship between the cause and effect, the results might not be valid outside the context of the specific study. Therefore, scalability from small, individual tasks to large tasks performed by teams need to be regarded. Similarly, the transfer from e.g. students to practitioners must be elaborated for different cases.

RESEARCH QUESTION 2.
How can requirements prioritisation techniques be characterised and compared?

The second research question was examined in two separate studies. The first study aimed at comparing requirements prioritisation techniques in two controlled experiments. In total, 46 academics participated in the experiments. The design was fixed, i.e. prepared and well-defined.

Controlled experiments are fixed in nature and apply to all four validity issues described earlier. *Internal validity* is considered by isolating the treatment from other influencing factors to ensure that the outcome is actually caused by the treatment. A typical example of threats to internal validity is that the groups given different treatments already differ from each other in one way or another. This was regarded by sampling the subjects based on pre-tests so that the groups' characteristics were as similar as possible and additionally the subjects were given treatments in different orders.

Conclusion validity was regarded by plotting the data and conducting appropriate hypothesis tests, which present significance of the results. The experiments were performed with a rather small and specific set of subjects. It increases the homogeneity of the subjects and thus the conclusion validity, although it reduces the *external validity* of the experiment since the subjects are not selected from a general population. The simplified setting and task might not be scalable to industrial usage. However, it is likely that the practitioners who are intended to use the techniques would perform similarly to the subjects (in this case mainly PhD students and master's students in their final year). Therefore, it would be appropriate to evaluate the techniques in industry.

Construct validity concerns the extent to which the experiment setting reflects the construct under study. In other words, we need to consider if we measure what we want to measure. The measures need to be defined and the treatments need to be applied carefully. Time-consumption is an objective and well-defined measure, which was tested using a watch. The subjects were aware of the measure, thus there may have been an interaction between testing and treatment. Ease-of-use is a subjective and well-defined measure, which was tested after the experiment using a questionnaire. No interaction between testing and treatment was present since the subjects were not aware of the test during the experiment. Similarly, no interaction between testing and treatment was present for the subjective measure of accuracy. However, the accuracy for a prioritisation technique is less well-defined, since there is no correct prioritisation key to compare with to determine accuracy. Construct validity is regarded by testing the same measures in two separate experiments. Performing the experiment with another set of requirements or another set of subjects would further increase the construct validity.

The second study was designed as an *archive analysis* (Robson, 2002) in which prioritisation data from requirements prioritisation assignments were used to evaluate the use of different measurement scales. The design was fixed, but differs from a controlled experiment since the subjects, as well as the researchers, were unaware of the usage of the data at the time of data collection. *Conclusion validity* was regarded since statistical tests were used when appropriate, and measures and treatments are considered reliable. However, the statistical power would be higher if more subjects were involved. *Internal validity* is less applicable in this case since the subjects were unaware of the analysis. Thus, threats such as learning effects and repeated testing are reduced. Threats to *construct validity* concerns whether the used measures actually reflects what we want to measure. We believe the presented measure of skewness to be well-defined and valid for comparing prioritisation results from different distributions, since it is based on the standard deviation.

Finally, the *external validity* can be discussed. Since the data are taken from a small-scale prioritisation task performed by students and PhD students it is difficult to generalise to an industrial setting. However, we believe that the study indicates that the presented measures are ready to be evaluated in industry.

3.4 Case Studies on Retrospective Analysis for Release Planning Decision-Making

The third research question was answered primarily by flexible case study methodology and the validity issues are mainly the same as presented in Section 3.2. This section describes the validity issues considered in the three case studies investigating the developed retrospective analysis method.

CASE STUDIES. A case study is conducted to investigate a single case within a specific time space and can be either fixed or flexible. The researcher typically collects detailed information on one single project, and different data collection procedures may be applied. Case studies differ from experiments in that the variables are not being manipulated, i.e. the case study samples from variables representing the typical situation. A case study is an observational study and may be easier to plan than a controlled experiment because it requires less pre-specification. However, it may be more difficult to interpret the result and generalise to other situations. Also, the effects of a change can only be assessed at a high level of abstraction and might not be possible to measure immediately (Wohlin et al., 2000). Case study methodology typically involves multiple data collection methods such as observation, interview and documentary analysis (Robson, 2002).

VALIDITY IN CASE STUDIES. The validity for case studies depends on the specific design. In this thesis, the case studies are mainly flexible. The collected data are primarily subjective and statistical methods are not applicable. The validity for flexible designs is discussed in Section 3.2. In addition, we can discuss *analytic generalisation* for multiple case studies

(Robson, 2002). The purpose of multiple case studies is not to gather a sample of cases so that generalisation to a population can be made, but to seek complements to the first study by focusing on an area not originally covered. In that manner it is possible to develop a theory which helps understanding other cases or situations. The strategy has similarities with performing multiple experiments in an attempt to replicate or complement earlier studies. Thus, the results may either confirm the theory or lead to revision and further development of the theory.

RESEARCH QUESTION 3.
How can retrospective analysis be used to evaluate and improve the release planning process?

The third research question was investigated in three different case studies in which the retrospective analysis method PARSEQ (Post-release Analysis of Requirements SElection Quality) was developed and applied. The method aims at finding improvements for the release planning process through retrospective analysis of already developed releases. A sample of requirements that were candidates for the investigated releases is re-estimated to find release planning decisions that would be made differently in retrospect. Those incorrect release planning decisions are investigated in a root-cause analysis where reasons for incorrect decisions are discussed, and improvements are suggested.

We used a flexible design and the three participating organisations have different characteristics, which required us to adapt the method to the different situations. The organisations were selected with respect to their differences, in order to discover limitations and possibilities of the method.

Description validity was regarded by taking extensive notes. In addition, charts and diagrams that were created during workshops were saved for further analysis. Since the method is based on the participants' knowledge and experiences, it is important to select the right people. In all cases several practitioners participated, such as product managers, project managers, chief developers, system architects, and users. They were carefully selected with respect to their experience of release planning for the investigated product. *Interpretation validity* was considered in two of the three cases since two researchers participated and could discuss the results afterwards to prevent misinterpretations. The results were also fed back to the participants for validation. *Theory validity* was handled by conducting multiple case studies with different characteristics. In addition, employees from different departments and in different roles participated in all studies. *Analytic generalisation* was regarded by seeking complementary cases so that different situations could be analysed. The case studies complement each other and give a comprehensive picture of the usage of the PARSEQ method.

4. Research Results

In this section, the main research results and contributions are summarised and plans for further work are described. Section 4.1 describes contribution C1-C3 which corresponds to the research questions RQ1-RQ3. Similarly, further research FR1-FR3, in Section 4.2, corresponds to future plans for RQ1-RQ3.

4.1 Main Contributions

This section describes the main contributions in the thesis C1-C3, corresponding to the three research questions. The results are discussed in relation to related research, some of which were presented in Section 2. Validity of the results is discussed in more comprehensive terms as a complement to the detailed validity discussion from a research design perspective in Section 3.

C1: Increased understanding of RE challenges in market-driven software development

The first research question aims at discovering challenges experienced by RE practitioners in software development. A flexible survey was performed with practitioners in industry. The paper reports on findings from interviews with 14 practitioners involved in RE at eight different software-developing companies. A large number of challenging issues were found, which were organised into twelve areas. Some of the challenges are also acknowledged by other sources. For example, the difficulty of writing understandable requirements is discussed by Al-Rawas and Easterbrook (1996). Further, the communication gap between marketing staff and developers is also described by Hall et al. (2002). In addition, the problem of implementing and improving RE in an organisation is also identified by Kauppinen et al. (2002).

Hence, several of the challenges discovered in the survey have also been identified in related research. However, some challenges have not been discussed in other surveys, probably because the challenges are special to market-driven development. For example, release planning is noticed as problematic for several of our participating companies, because it is often based on uncertain estimates of cost and value. All participants discuss requirements prioritisation, although most of the companies use an *ad hoc* approach based on grouping requirements. Similarly, managing the constant flow of incoming requirements suggestions, and handling the balance between eliciting requirements from potential users and inventing new ones in-house were referred to as challenging.

Challenges confirmed by other sources increase the credibility of the results, and new ones help us increase the understanding of the RE challenges experienced by practitioners. The sampling was made with respect to the differences between participants since we wanted to discover an as broad spectrum of challenging issues as possible. It is possible that other challenges would appear if other companies participated. Thus, the picture of industrial RE challenges we provide is not a universal one. However, it has suited its purpose of increasing the understanding of the area, and helping to find research areas in need of further investigation. The area of requirements prioritisation and release planning decision-making was selected for the continued research because it was one of the four areas which were discussed by all participants and it received the highest number of quotations in the interviews.

C2a: Report on characteristics of different requirements prioritisation techniques

The contribution to the second research question is divided into two parts: C2a and C2b. The first one is investigated in experiments, evaluating the differences in time, ease of use, and accuracy between three requirements prioritisation techniques. The results suggest that Pair-wise comparisons with tool-support (Telelogic, 2006) is the fastest of the three investigated techniques, and the Planning game is the second fastest. The two mentioned techniques do not differ regarding ease of use. The manual Pair-wise comparisons technique is the most time-consuming and least easy to use among the investigated techniques. The accuracy of the prioritisation results does not differ among the techniques.

These results contradict some earlier work. In Karlsson et al. (1998), a technique similar to the Pair-wise comparisons technique is found to be superior to a technique similar to the Planning game regarding ease of use and reliability of results. However, Pair-wise comparisons is the most time-consuming technique in their evaluation, which is also true in our case. Our results are supported by Lehtola and Kauppinen (2006) who discovered in their case study that pair-wise comparisons were difficult and time-consuming to perform, especially with more than 20 requirements. Some users also argued that pair-wise comparisons are pointless and it would have been easier for them to just select the most important requirements without comparisons. On the other hand, dividing requirements into three groups, as is done in the Planning game, is often used in practice (Lehtola and Kauppinen, 2006; IEEE, 1998; Karlsson, 1996). Techniques based on grouping and ranking, such as the Planning game, may be more efficient to introduce than the Pair-wise comparisons, since grouping is often already used in practice.

In summary, two studies confirm that pair-wise comparisons is a time-consuming technique (Karlsson et al., 1998; Lehtola and Kauppinen, 2006). Therefore, we regard the results on time-consumption as trustworthy. However, the results regarding ease of use is confirmed in one study (Lehtola and Kauppinen, 2006) and contradicted in another (Karlsson et al., 1998). Therefore, further studies are needed to investigate this difference in results before validity can be assured.

C2b: Evaluation of measurement scales in requirements prioritisation

The second research question is also investigated in an archive analysis, examining the decision-support provided by different requirements prioritisation techniques using different measurement scales. The measurement scales relevant to requirements prioritisation are the ordinal scale and the ratio scale, which are further described in Part II and in (Fenton and Pfleeger, 1997).

Results from prior prioritisation exercises were re-examined in an archive analysis. Four different data sets, with 36 data points in total, resulting from prioritisation with a ratio scale technique were investigated. The purpose was twofold: to investigate the skewness of the different ratio scale prioritisation results, and to compare the cost-value approach for ordinal and ratio scale data. The paper presents a measure for the skewness based on the standard deviation from a baseline distribution. The measure indicates that some of the subjects tend to get a more skewed distribution, i.e. they use the extreme values on the ratio scale more than others. The subjects expressed that one reason for using modest values is lack of domain knowledge. The measure can be used to evaluate in which situations it is worth the added effort of using the ratio scale compared to the ordinal scale.

The evaluation of the cost-value approach compares cost-value diagrams drawn from ordinal scale data to diagrams drawn from ratio scale data. It indicates that the ordinal cost-value diagram agree substantially to the one based on ratio scale data. Thus, decisions based on ordinal scale data would be substantially similar to decisions based on ratio scale data.

The results speak in favour of using the ordinal scale, at least in situations when domain knowledge is weak and it may be sufficient with ordinal scale data. The cost-value approach can then be used as decision-support. Lehtola and Kauppinen (2006) support this view by describing that some practitioners found it difficult to estimate which number to give to factors when using a ratio scale technique. Further, they conclude that prioritisation techniques are valuable for putting a set of requirements in order, and using the results as a basis for discussion.

The presented approaches to compare ordinal and ratio scale data are novel and we need more data to confirm the conclusions. Industrial usage is needed to validate whether the ordinal scale can bring the same decision-support for practitioners as the ratio scale.

C3a: Method for retrospective analysis of release planning decision-making

The contribution to the third research question is divided into two parts: C3a and C3b. The third research question investigates how retrospective analysis can be used to improve the release planning process. A method called PARSEQ was developed for the purpose of analysing the release planning process and improve the requirements selection quality in a structured manner.

The method is evaluated in three separate industrial case studies with different characteristics. The first case examined a small software developer, the second investigated an in-house software project, and the third examined product line development for an embedded software product. In the first case a handful of improvement suggestions for the release planning process were found. The second case was found to have made successful release plans for the product and the study focused on the positive experiences from the project. In the third case, a large number of root-causes and improvement suggestions were found.

The application of the PARSEQ method differed between the cases. The first one was supported by a requirements management (RM) tool and a ratio scale prioritisation technique was used. Since we wanted a faster approach, not depending on a commercial RM tool, the ordinal scale techniques were considered. The studies for RQ2 indicated that the ordinal scale seemed sufficient for our purposes, and therefore the other two cases used a more agile approach based on the Planning game and ordinal scale cost-value diagrams.

Retrospective analysis has shown fruitful in other areas such as project management (Kerth, 2001), knowledge management (Rus and Lindvall, 2002), and agile development (Collins and Miller, 2001). Our results indicate that it is also successful in finding improvements for the release planning process. The three case studies are performed at different organisations with different characteristics. The method seems feasible for the investigated cases and therefore it is likely that it will work in other situations as well, although further cases need to be investigated to find the possibilities and limitations of the method.

C3b: Tool-support for the retrospective analysis method

Based on the experiences from the first two case studies, a tool was developed with the purpose of making the process more efficient and increase possibility of visualisation. The tool handles all steps from importing a sample of requirements to exporting process improvement suggestions. The tool uses a number of windows to guide the user through the process. The re-estimation can be performed with one of three different requirements prioritisation techniques: the Planning game, the $100 method, and the Pair-wise comparisons. Two criteria of ones own choice can be entered: one to maximise and one to minimise, e.g. value and cost. After re-estimation, the tool illustrates the results in a cost-value diagram, which is then used for analysis. The discussion regarding root-causes and improvement suggestions can be documented in a root-cause matrix, which in turn can be exported along with the cost-value diagram from the tool.

The tool was used and evaluated in the third case study. The evaluation shows that it did speed up the process and decrease manual labour. It was valuable to be able to select prioritisation technique and criteria during the workshop. It also provided good visualisation support through the automatically generated cost-value diagram. However, some drawbacks were also found, such as lack of support for distributed workshops. Further development is needed before it can be used as support in all steps of the method.

4.2 Further Research

This section describes how the research can be continued and evolved in the future. All included papers have possibilities of deeper investigation, which is further detailed in the different parts. This section is arranged in three sub-sections, describing further work for each research question.

FR1: Increasing survey sample with focus on diversity and good experiences

In order to provide a more comprehensive picture of practitioners' RE challenges, it would be valuable to add further interviews with people in other organisations. Although the sample in the conducted qualitative survey is broad, additional medium-sized organisations would further extend the sample. Organisations that develop embedded systems, as well as products sold on consumer markets would further increase the range of the sample. The agile development approaches are gaining land and further experiences from agile and incremental development would be valuable.

The performed study focused on challenges in RE. However, it could be even more valuable for practitioners to report on good experiences from successful projects. Therefore, future studies could focus on projects and organisations which can share their knowledge and demonstrate encouraging examples of solutions to the stated challenges. A large scale study of that kind could end up in a guidebook, where practitioners could recognise challenges and find examples of possible solutions based on industrial experiences.

FR2: Industrial validation of requirements prioritisation techniques

The results from the investigation on requirements prioritisation techniques would benefit from industrial validation. It is not certain that the results achieved in the papers are valid for industrial use since the scale and domain are different. The number of requirements investigated is small compared to most situations in a real project and the domain of high-level requirements for mobile phones is a simplified representation of real requirements. An industrial case study could involve a combination of the investigated techniques, i.e. to use a simple technique (such as the Planning game) for assigning requirements in groups and then use a more rigorous technique (such as Pair-wise comparisons) for the requirements that need more detailed evaluation.

There are other techniques that could be compared in further experiments, such as Wiegers' method and the $100 method. Both are based on a ratio scale but Wiegers' method takes several criteria into consideration, while the $100 method focuses on one criterion at the time. The criteria in Wiegers' method are customer value, penalty if the requirement is not implemented, cost of implementation, and risks. An interesting procedure would be to compare Wiegers' method to the cost-value approach, where cost and value are estimated with e.g. $100 method, to evaluate which one gives more valuable support for release planning. In addition, time-consumption and ease of use can be measured.

To further validate the usage of different measurement scales for requirements prioritisation, data sets from industrial requirements prioritisation sessions may be used. It would then be possible to investigate whether some people use the more extreme values on the ratio scale, while others are more modest, also in industrial requirements prioritisation. Industrial requirements prioritisation data could also be used to further evaluate the usage of the ordinal cost-value diagrams. It may be possible to set up a case study in industry to evaluate if the practical decision-support achieved by the ordinal scale prioritisation techniques is sufficient. Interviews could then reveal the practitioners' opinions after using different techniques.

FR3: Possibilities and limitations for the PARSEQ method

The PARSEQ method could be applied in additional organisations to examine its possibilities and limitations. The method need to be adaptable to different situations such as different development approaches, different project types, and different product types. Therefore, organisations and projects with different characteristics need to be involved. If a large number of case studies is performed, the method could be used as a ground for finding general improvement areas to the release planning process. It may be possible to see patterns between certain organisational characteristics and certain process improvements. In that case, general recommendations could be developed regarding release planning process improvements.

The tool support for the method also needs more evaluation and improvement. Improvements are needed e.g. to be able to perform PARSEQ in a distributed manner. Modifications can be developed as part of a Master's student project. Thereafter, the tool needs industrial validation in further case studies.

Available resources for future research

As discussed above, this research can be carried on in a number of ways. To assist others who aim at investigating this area we recommend the following available resources. This thesis is available online along with the included publications at http://serg.telecom.lth.se/research/publications/. In addition, the design and other material used in the experiments in Part II can be found at http://serg.telecom.lth.se/research/packages/ReqPrio/. The PARSEQ tool and source code, along with guidelines and the development report, can be downloaded from http://serg.telecom.lth.se/research/packages/ParseqTool.

PART I: REQUIREMENTS ENGINEERING CHALLENGES

Paper 1. Requirements Engineering Challenges in Market-Driven Software Development - An Interview Study with Practitioners.

Lena Karlsson, Åsa G. Dahlstedt, Björn Regnell, Johan Natt och Dag, Anne Persson

Accepted for publication in Information and Software Technology: Special Issue on Understanding the Social Side of Software Engineering, Qualitative Software Engineering Research, 2007.

I

Abstract

Requirements engineering for market-driven software development entails special challenges. This paper presents results from an empirical study that investigates these challenges, taking a qualitative approach using interviews with fourteen employees at eight software companies and a focus group meeting with practitioners. The objective of the study is to increase the understanding of the area of market-driven requirements engineering and provide suggestions for future research by describing encountered challenges. A number of challenging issues were found, including bridging communication gaps between marketing and development, selecting the right level of process support, basing the release plan on uncertain estimates, and managing the constant flow of requirements.

1. Introduction

This paper reports on results from an industrial qualitative survey[1], focusing on current practice and challenges for market-driven requirements engineering (RE) in Swedish software development organisations. Market-driven software can be combined with hardware in embedded systems or sold as COTS (Commercial Off-The-Shelf) products. Before problems related to inefficient RE can be properly addressed, more research is needed to better understand the challenges faced by the software industry. The purpose of the research is to discover and describe RE challenges found in industry today, in order to increase the understanding of the area of market-driven RE. The study also aims at discovering proposals for future research in the area, which is important since most available research focuses on the traditional customer-specific manner of software development. In addition, the study complements existing RE surveys, since few of them have focused on the specific challenges found in market-driven software development. The main research question investigated in this paper is: *Which RE challenges do market-driven software development companies face?*

The study focuses on market-driven software development, which is gaining increased interest in the software engineering community compared to development of customer-specific systems. This is due to the emergence of the market for COTS or packaged software (Carmel and Becker, 1995; Sawyer, 2000a). Software is an essential part of numerous commercial products today, and hence, a growing number of companies are involved in market-driven software development. Products of all kinds, such as mobile phones, automobiles, aircrafts, toys, and games, contain software. Market-driven software products are vended on an open market and there is a large range of potential customers, thus diverse requirements need to be considered. Market-driven products are often developed in several consecutive releases for which competition is high. The characteristics of market-driven RE differs from the characteristics of customer-specific RE in several ways, as further explained in the subsequent section on related work.

There are several surveys that concern or include RE issues (Chatzoglou, 1997; Curtis et al., 1988; El Emam and Madhavji, 1995;

[1] By *qualitative survey* we refer to a field study including a series of interviews with practitioners

Hall et al., 2002; Hofmann and Lehner, 2001; Lubars et al., 1993; Nikula et al., 2000). However, none of these focus primarily on market-driven development. Furthermore, in most of these surveys, the studied projects and organisations are mainly large, both in terms of the number of persons and requirements involved, and in terms of the duration of the projects. This qualitative survey complements the mentioned ones and provides a characterization of market-driven RE from the perspective of small- and medium sized companies, as well as fairly new ones.

In this survey, fourteen persons from eight different companies participated in interviews. After seven interviews, a short paper (Paper 8) was presented at an international workshop. In addition, a focus group meeting involving RE experts was held halfway through the study to get feedback on the challenges found so far. Semi-structured interviews were held with each interviewee. Each interview was recorded and transcribed for the analysis, which was supported by the qualitative data analysis tool Atlas.ti (Atlas.ti, 1997).

The paper includes a description of the companies involved, focusing on company facts, typical projects and development processes. The result of the study is a set of issues that may increase the understanding of the challenges faced by market-driven organisations, as well as indicate the direction of further research. In addition, we discuss our experiences from using a qualitative research approach.

The remainder of the paper is organised as follows. In Section 2, related work is presented. The research method is described in Section 3. Section 4 presents the results of the study and ends with a summary. Section 5 discusses the results and relates the findings to earlier work. It also discusses the threats to validity and our research experiences. Section 6 concludes the paper and presents some ideas for further work.

2. Related Work

This section describes some earlier work related to the topics of market-driven RE and RE surveys. The included references are the ones found most important to our study, either with regard to the research design or to the research results.

2.1 Market-Driven Requirements Engineering

Although there are many commonalities between market-driven and customer-specific development, the literature argues that there are also differences of such kind that these need to be made explicit (Sawyer, 2000b). The major differences include the characteristics of stakeholding and schedule constraints (Sawyer, 2000a) as well as release planning and managing the constant flow of new requirements (Carmel and Becker, 1995; Honour, 1995; Potts, 1995).

In market-driven projects, there is no distinct and defined set of users (Sawyer, 2000a). There are mainly potential users, an imagined group of people who may fit into the profile of an intended product user. Eliciting requirements from this group of users and customers is one of the major distinguishing characteristics between market-driven and customer-specific RE (Deifel, 1999; Potts, 1995; Sawyer et al., 1999). This is mainly managed through marketing, technical support, user groups and trade publication reviewers (Carmel and Becker, 1995).

Often, requirements are *invented* by the developers (Potts, 1995), e.g. based on strategic business objectives, domain knowledge and a product vision. The development organisation is the primary stakeholder, and hence it decides which requirements to use in the next release. Nevertheless, in order to keep, or increase, market shares, the requirements that satisfy most customers need to be selected. This further emphasises the role of marketing in the market-driven development situation (Deifel, 1999).

Within market-driven development organisations, *time-to-market* is described as a survival attribute (Novorita and Grube, 1996; Sawyer et al., 1999). If the product is not released to the market on time, there is a risk of losing market shares to competitors. As a consequence, the release dates are kept fixed and requirements of a lower priority may hence be excluded from the current release in case of a delay (Carlshamre, 2001). Release planning, aiming to select a set of requirements for the next release that maximises customer value taking into account the available resources (Carlshamre, 2002), is therefore a major challenge (Sawyer et al., 1999). However, there is usually a steady stream of new requirements, improvements suggestions, complaints and bug reports suggested by existing customers and users of previous releases. Therefore, effective prioritisation and cost/impact assessment is needed to support the release planning task (Carlshamre, 2001; Karlsson, 1998; Sawyer et al., 1999).

Moreover, the RE process needs to include procedures to capture and preserve this steady stream of requirements (Higgins et al., 2003).

2.2 Requirements Engineering Surveys

There are several surveys that concern or include RE related challenges. The classical article by Curtis et al. (1988) mainly involves large system development projects, both customer-specific and market-driven ones. Although their survey does not focus solely on RE, it identifies three major challenges which all can be related to RE: the thin spread of application domain knowledge, fluctuating and conflicting requirements, and communication and coordination breakdowns. A few years later, Lubars et al. (1993) published their field study on requirements modelling. Their study also includes both customer-specific and market-driven projects, as is also the case in the paper by Curtis et al. (1988). The challenges found in (Lubars et al., 1993) is well in line with the ones in (Curtis et al., 1988) and include e.g. vaguely stated requirements, missing requirements, requirements misunderstandings, the lack of definitive requirements from outside the development organisation, lack of customer contact, and frequently changing requirements.

Next, two studies on RE practices are published, which both also report on experienced problems (Chatzoglou, 1997; El Emam and Madhavji, 1995). In the paper by El Emam and Madhavji (1995), the challenges identified are more related to project management issues such as performing the appropriate activities, deciding when to stop particular activities, ensuring an adequate level of user participation, and selecting capable personnel for the key roles in RE. The survey by Chatzoglou (1997) mainly includes projects of a customer-specific characteristic. The challenges presented are e.g. lack of enough resources for a successful completion of the RE process, and not adequate quality of tools and techniques employed in the RE process.

Later on a study on RE challenges was presented by Kamsties et al. (1998). Unlike the field study by Curtis et al. (1988), this study includes small- and medium sized enterprises. However, both customer-specific and market-driven projects were represented. Kamsties et al. (1998) agree to some extent with Curtis et al. (1988) and Lubars et al. (1993) and identify unclear and incomplete requirements as a common challenge. Other challenges mentioned are: complexity of requirements documents, lack of documented requirements in market-driven projects, and cases when implementation of new requirements can cause unforeseeable interaction with requirements already implemented.

Two more recent papers on RE challenges are written by Hofmann and Lehner (2001) and Hall et al. (2002). The first survey aims at establishing a clear link between RE practices and performance, and includes both customer-specific and market-driven projects. Also in this paper, changing requirements are identified as challenging. They also support the findings by El Emam and Madhavji (1995), concerning selecting capable personnel, since they found that there is a lack of involvement of technically knowledgeable stakeholders when defining the initial requirements. Other challenges identified in (Hofmann and Lehner, 2001) are unrealistic requirements posed by marketing, and perceived ad hoc RE processes. Moreover, they have also shown that approximately 15% of the project effort was spent on RE activities. The paper by Hall et al. (2002) emphasises that most requirements problems are organisational rather than technical. Hall et al. (2002) also discuss a relation between the maturity of the companies and requirements problems found.

Finally, there are a number of RE surveys that are not focusing on challenges (Breitman et al., 1999; Nikula et al., 2000) and others which investigates certain kinds of challenges, e.g. the survey by Damian and Zowghi (2003) which describes challenges caused by stakeholders' geographical distance.

In general, none of the presented RE surveys have a primary focus on market-driven development, although some includes both market-driven and customer-specific organisations/projects (Curtis et al., 1988; Hofmann and Lehner, 2001; Kamsties et al., 1998; Lubars et al., 1993). Hence, none of these gives an overview of challenges faced specifically by market-driven development organisation regarding RE issues, despite the fact that there are pronounced differences between customer-specific and market-driven RE. Therefore, there is a need for an investigation of the RE challenges faced by market-driven development organisations.

3. Research Methodology

The study was carried out using a qualitative research approach. Qualitative research aims to investigate and understand social and cultural phenomena in the context where they exist. Furthermore, qualitative research aims to place the studied phenomena in a holistic framework and suggests that social and cultural phenomena are best investigated by studying people's actions in and verbalised thoughts about the social context in which they act (Myers, 1997). Qualitative research methods are useful when the purpose is to explore an area of interest, to obtain an overview of a complex area, and to discover diversities and variety rather than similarities (Robson, 2002). It is also preferable to use a qualitative approach when the aim is to improve the understanding of a phenomenon where little is known. This is due to the fact that it focuses on gaining in-depth information (Hoepfl, 1997).

The purpose of the study presented in this paper is to gain in-depth understanding of the nature of requirements engineering within market-driven software companies, specifically focusing on exploring the challenges software development companies in this sector face. The aim is also to provide a basis for formulating hypotheses for future research. Due to this explorative nature of the study a qualitative approach has been considered suitable.

The study is mainly built on semi-structured interviews with a high degree of discussion between the interviewer and the interviewee, complemented with a focus group meeting. This approach has enabled the researchers to gain in-depth understanding e.g. of the concepts/terminology used by the companies. This proved to substantially facilitate data analysis. For example, concepts such as requirement, process and method had different meanings among the companies. An alternative approach could have been to use a questionnaire. The drawbacks of such an approach would be that the assumptions of the researchers in that case govern the questions asked and the terminology used. The explorative element of the study would, hence, be difficult to achieve. Furthermore, a questionnaire does not provide the opportunity to discuss what questions and concepts mean and to explore new views on the area that may appear during a face-to-face interview.

Quantitative methods are sometimes claimed to be "better" than qualitative methods with the argument that they provide objective measurement and enable replication of studies, as opposed to qualitative methods that build on inherently subjective interpretation. However, we do not consider debating which type of method is "better" than the other to contribute constructively to science. Instead we need to take a more general view of long-term research (Figure 1). The cycle of long-term research entails 1) discovery of hypotheses/theories and 2) justification/proof/validation of hypotheses/theories. The discovery phase is followed

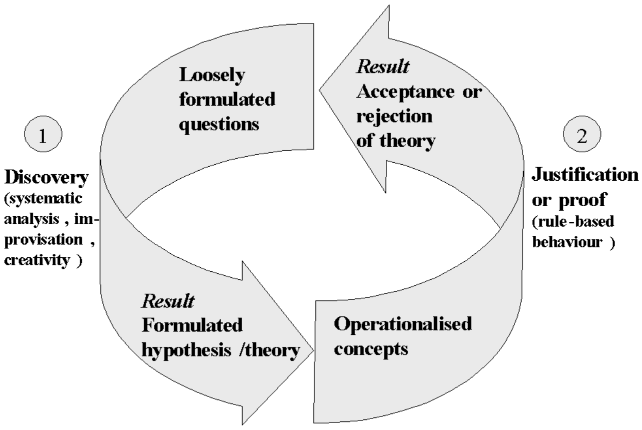

Loosely formulated questions
Result
Acceptance or rejection of theory
1
2
Discovery
(systematic analysis , improvisation , creativity)
Justification or proof
(rule-based behaviour)
Result
Formulated hypothesis /theory
Operationalised concepts

Figure 1. *The long-term cycle of research*

by justification and then new theories or elements of theories can be discovered, which need justification, etc.

In the study reported in this paper, we have discovered a number of challenges to RE in market-driven software development companies. These challenges can be seen as hypotheses or be the basis on which new hypothesis, studies and experiments can be formulated and carried out.

The study consists of three steps, which are described in the following. Each step is divided into three phases: planning, data collection and analysis (Figure 2).

3.1 Step 1 - Interview study (part 1)

Planning. The first part of the study involved a brainstorming and planning meeting to identify different areas of interest and to plan the study. We used a combination of maximum variation sampling and convenience sampling since we selected companies with as different characteristics as possible within our industrial collaboration network (Patton, 2002). We aimed at a variety of companies, with respect to size, type of process, application domain and type of product. We started with five companies, in which seven persons were interviewed. Interviewees in different roles were involved in order to collect different viewpoints and perspectives on the nature of RE.

The interview instrument was designed with respect to the different areas of interest and with inspiration from other RE surveys (El Emam

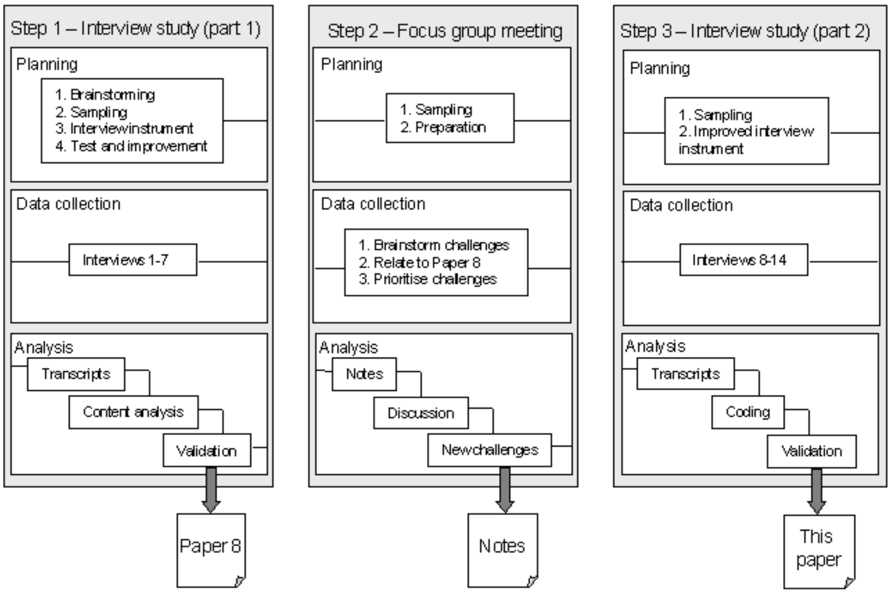


Figure 2. *The research procedure used in the interview study*

and Madhavji, 1995; Lubars et al., 1993). To test the interview instrument, two pilot interviews were carried out. Some questions were clarified and the structure of the interview instrument was improved before interviewing proceeded. A summary of the interview instrument is available in the appendix, Table A.

Data collection. The study uses a semi-structured interview strategy (Robson, 2002). All interviews were attended by one interviewee and three interviewers, one of which was responsible for the interview process. The other two took extensive notes in order to gather as much information as possible. All interviews, varying in length from 90 to 150 minutes, were also recorded. Transcripts of all interviews were made in order to facilitate and improve the analysis process. The transcripts varied between 7 and 23 pages in size.

Analysis. The *content analysis* (Patton, 2002) involved marking and discussing interesting sections in the transcripts. The interviewers examined the transcripts from different perspectives and searched for explicitly stated or concealed RE challenges. Another researcher, who did not attend the interviews, also analysed the transcripts to enhance validity. The results from Step 1 are presented in Paper 8.

3.2 Step 2 - Focus group meeting

Planning. The aim of the focus group meeting was to obtain feedback from industrial RE experts on the eleven RE challenges found so far, and to find additional ones. Therefore, we selected two participants who were interviewed in Step 1, and three new participants that did not attend the interviews.

Data collection. The meeting started with a brainstorming session where the five industrial RE experts wrote down challenges from their own experience on post-it notes. The session was facilitated by one of the five researchers present. Next, the industrial experts mapped their post-it notes to the challenges found in Step 1. Some post-it notes could not be mapped to a challenge and therefore three new challenges were formulated. All fourteen challenges were prioritised based on a simple scheme; every participant had ten votes each to distribute among the challenges they found most important.

Analysis. The five researchers took extensive notes, which were discussed and composed after the meeting. No particular analysis was made of the notes at this stage. Instead, the notes were used as input to the analysis of the third step, the second part of the interview study.

3.3 Step 3 - Interview study (part 2)

Planning. We continued with the sampling strategy developed in the initial step of the study, and interviewed seven persons at five software companies. The new companies included both large and small organisations[2]. Both very large and very small projects appear in the new

[2] Due to lack of organizational information regarding annual turnover our definition of small-, medium-, and large organizations is approximated. For an elaborated definition, see e.g. http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf

companies and more attention was also paid to developers of embedded systems in order to broaden the scope.

The interview instrument was adjusted. Some questions were enhanced with more detail, while others were given a more open-ended structure.

Data collection. The semi-structured interview approach was continued. The interviews varied between 60 and 120 minutes in length. One interview was attended by two interviewees due to lack of time. All interviews were conducted by one to three interviewers and were recorded on tape. Notes were taken in the same manner as during step 1 and each interview was transcribed before analysis.

Final analysis of all data. Since we sought a comprehensive view of the complete data set, the data from the first part of the study was re-analysed together with the data from the focus group meeting and the second part of the interview study. The analysis was supported by the qualitative data analysis tool, Atlas.ti. As the data grew rather extensive (more than 160 pages of transcribed text), tool support was necessary in order to maintain structure, as well as to facilitate cooperation between the researchers, who worked at different geographical locations.

In the final analysis we used the eleven challenges that emerged in the first set of interviews together with the additional three that were discovered during the focus group meeting as predefined categories/codes (Glaser and Strauss, 1967). In addition, one more category emerged as our understanding of the data increased through discussing and working with the data. Thus, a total number of 15 categories were found (Table 1). Each category was described with quotations and examples, in order to develop a common understanding of each category. Within each category we found more detailed challenges, which were further analysed.

With reference to Grounded Theory (Glaser and Strauss, 1967) the process throughout the three steps of the study reached what can be described as theoretical saturation. The number of new categories that emerged from each step became smaller and smaller, from eleven to one. Apart from one more category, the third step mainly provided more detail and confirmed previous findings. This is illustrated in Table 1.

The interview transcripts were divided between two researchers for coding. The transcripts were analysed and interesting quotations were marked with one or more of the 15 categories. Afterwards, the researchers

Table 1. The 15 categories used during analysis

Challenges that emerged from the interview study (part 1)	**Challenges that were added after the focus group meeting**	**Challenges that emerged from the final analysis of all data**
Design in the requirements	Requirements for requirements	Non-functional requirements
Simple techniques and tools	Organisational culture	
Requirements repository	Tool integration	
Requirements bundling		
Requirements overload		
Requirements changes		
Market-driven/Technology-driven		
Organisational stability		
Process		
Release planning		
Gap between marketing and developers		

switched interviews to validate that coding was made in a consistent manner. It was also possible to add more categories at that point. Each researcher kept a file in the tool that contained the coding made by that researcher. When coding was finished, the two individual files were merged into one within the analysis tool. For the analysis, all related transcript quotations for each category were compiled and printed in order to compile the data into a readable format. The results from the analysis are found in Section 4.

In order to further ensure the quality of the coding process, a third researcher that did not attend the interviews compared the transcripts with the results given in this paper. Quotations in the results were traced to their source in the transcripts to check translation and to ensure that the interpretations converged. Due to traceability problems, caused mainly by translation from Swedish, two quotations were not recovered in the extensive transcript material and were therefore removed from the results. One quotation was changed to a descriptive explanation, since it was difficult to translate without losing the essence of what had been said.

In six quotations, single words were erased or changed to obtain a better language structure.

4. Results

This section presents the challenges discovered during the analysis of the interviews and the focus group meeting. The twelve following subsections present and discuss one or several challenges corresponding to one or several of the categories presented in Table 1. Company and interviewee characteristics are available in the appendix, Table B. The challenges are subsequently elaborated and explained using excerpts from transcripts and interpretations from coding and analysis.

4.1 Simple techniques for basic needs

The Rational Unified Process (RUP) (Jacobson et al., 1999) was mentioned by several of the interviewees as being the development process used. For example, the employees at Company F have used RUP and Rational tools for a few years and they are generally satisfied with their performance. However, some problems were mentioned, e.g. that RUP lacks support in the handover between requirements engineers and designers. Another issue is that RUP is a use case driven approach, while the company wanted to work in a feature-driven manner. This adds some workload as the traceability between features and use cases needs to be maintained.

At Company E, RUP and Rational tools were tested, but the employees had trouble accepting the process. "It just gets too complex" was the comment from the managing director. At Company B, RUP was under consideration but it was difficult to find the necessary time and effort for it to be introduced in the organisation. The project manager's comment was "We do not have time to implement RUP and those tools". The product manager at the same company said that they want a simple tool, "that you do not need a thick manual to understand". The project manager requested an RE tool, such as DOORS, to support RE in the projects. Thus, the need for tool support seems to vary depending on organisational role. It also seems to be the smaller organisations that have trouble finding processes and tools that suit their needs. During the focus group meeting, a lack of tools that support a flexible process was discussed. The difficulty to integrate different tools (e.g. RE tools and testing tools) was also an issue.

Some companies use the Unified Modelling Language (UML) (Fowler and Scott, 2000) for specification. The opinions about UML vary among the interviewees. At Company C, UML is regarded as easy for customers to understand, while at Company E it is stated that "you cannot take the models to someone without a masters degree" and therefore it is not a good means for communicating with customers. The difference in views regarding UML is probably due to different customer characteristics. The software developed by Company C is intended for developers and product managers, while the software developed by Company E is intended for managers and developers of business processes. Company H has tried a use case methodology but it was abandoned since they perceived that it had shortcoming as a feedback instrument in customer communication. On the other hand, they had positive experiences with prototyping to get customer feedback. They stated that they need techniques and methods that are easy to understand in order to get customer feedback.

4.2 Communication gap between marketing staff and developers

At the focus group meeting, the lack of co-operation between different parts of the organisation, especially marketing and development, was recognised by most participants. One participant stated, "If these two departments communicated, we would not even have to specify the requirements. […] If everyone has the same goal and vision, then everyone works in the right direction." At Company B, the project manager suggests that one way to solve the lack of communication between marketing and developers is for marketing staff to learn UML as it could function as a common language for communication of requirements.

In Company A, the communication gap between marketing and development was also obvious. The marketing department's view of specifying requirements was to write down ideas for future functionality, while the developers expected clear and detailed requirements that could be used for design. Nobody took responsibility for the specification and analysis of requirements. At Company F, the problem was managed by a systems management group, which acts as a "mediator" between marketing and developers as well as re-formulates the one-line requirements from marketing into designable requirements for the developers. This is a good solution according to the system manager at Company F, as product management does not have as detailed knowledge as systems management.

Another communication problem concerns providing the sales department with sufficient information before customer meetings. At Company B it had been the case that sales staff promised functionality to customers before confirming its feasibility. At the focus group meeting this was mentioned as a challenge and it was suggested to have the sales department communicate with requirements engineers to solve the problem.

Another indication of communication gaps between departments is the different opinions as to what constitutes a "good requirement". At Company B, the product manager thinks that a good requirement is one that yields high revenue with low effort, and that is possible to sell. The managing director at Company C expresses a similar view. However, according to the developers in the same organisations, good requirements are independent, testable, clear, and not conflicting. It seems that the notion of a "good requirement" vary depending on organisational role.

4.3 Writing understandable requirements

Most of the companies use Natural Language (NL) to document their requirements. This is sufficient according to most interviewees since it is the customers' language and customers need to understand the requirements. However, one of the interviewees at Company H states that different stakeholders use different vocabulary, or "they use the same word but with a different meaning". This makes NL problematic.

Producing well-formulated requirements is an issue, since many of the interviewees find it difficult to understand requirements. The head of developers at Company C considers it as one of the major challenges in carrying out his daily tasks. However, the managing director at the same company does not consider this to be a major problem. The difference in views seems to be related to the role within the organisation. It is possible that marketing staff do not need the deeper understanding of requirements, and can therefore cope with less well formulated requirements. However, developers may need more detail to manage implementation. Many interviewees put effort into increasing the clarity and understandability of requirements, mainly through group discussions. The head of developers at Company C said, "It is not possible to write requirements completely, clear as a bell. You always need to discuss with others."

4.4 Managing the constant flow of new requirements

Market-driven product developers need to deal with a constant flow of new product requirements. It is considered important to collect all these new and potentially valuable ideas, derived from e.g. customers, users, developers, and support personnel. However, there needs to be efficient means to manage them. Several of the companies use a repository or tool to store incoming requests and these tools are then used to search for interesting, and valuable requirements to implement in the next release of the product. In Company B, C, and G, the same repository is used for requirements and Trouble Reports (TR), since both compete for the same resources. Even though a repository ensures that no good ideas are lost, it poses a number of other challenges.

Firstly, Company A, C, and D allow their customers and other stakeholders to enter requirements and ideas by themselves. However, this requires various efforts afterwards in order to improve the understanding of the statements and to identify duplicates amongst the requirements.

Secondly, Company A, which has a mature product, has had severe problems with overload in the requirements database, since more requirements are added than could be managed. This has resulted in difficulties when prioritising the requirements for the next release, since there were thousands of requirements to consider. The problem was temporarily solved by creating a list that included requirements considered the most important for the next release. There is, however, an apparent risk that some rejected requirements would have been more important after all.

Thirdly, the project manager at Company D emphasises the importance of giving feedback to all suggestions from stakeholders. Otherwise they may feel neglected and "if they knew [their suggestion] was sent into nowhere, they would stop suggesting". Thus, it is highly important that customers know that all suggestions are taken into consideration. However, giving feedback to the constant flow of new requirements from a large number of customers requires a large amount of effort.

4.5 Requirements volatility

Requirements change for a number of different reasons such as e.g. when the market fluctuates, problems are found during coding or reviews, and resource constraints change. Some interviewees fear volatility, while others welcome it. For example, at Company B the project manager said, "it takes time before the product is stable, and that is what you want to reach". At Company D on the other hand, volatility is accepted as a fact of software development. The project manager states, "[The customers] will change their minds when they see it, they change their mind in any case". Therefore they use an agile development approach. The project manager also said that "Code is […] cheap to throw away, […] it's almost for free". Therefore it is possible to build imperfect code and show it to the customer for feedback. This approach is also used at Company C, where a function is often developed to 60-90% so that feedback can be received earlier without having to spend 100% of the planned resources. The developer said, "it's not worth finishing a requirement to 100% because change requests will come […] so we accept it earlier and wait for comments". The trade-off between stability and volatility was also discussed at the focus group meeting where one participant said: "the environment changes while we want to freeze the specification and strive for stability in the projects". Another participant requested requirements models that are easy to change.

4.6 Requirements traceability and interdependencies

All participants involved in the study acknowledge the existence of relationships and interdependencies among requirements. However, their perceptions with regard to its importance and impact on the development work differ substantially between companies. Influencing factors seem to be how elaborate and well-defined the RE process is, but more importantly the size of the project and the complexity of the product.

In six of the companies, relationships and dependencies between requirements are usually used for bundling related requirements to be implemented together. Bundles are made in order to increase efficiency during implementation, e.g. because the requirements relate to the same part of the system or should be implemented by the same developer.

Company F has a more rigorous way of dealing with requirements interdependencies. They use a so-called *anatomy plan* to describe the functional structure of the software. It is a description of how the functionality of the system is related, i.e. dependencies between functions. It is mainly used to describe the order in which the functionality of the system should be implemented. Company F states that even if it is difficult, functional dependencies can be managed in this way. A major problem is to deal with what they call dynamic interdependencies, i.e. quality characteristics or non-functional requirements, which influence a larger part of the functionality or other characteristics of the system. The problem is not only to know that the interdependency exists but also to figure out how the requirements affect each other and how this can be dealt with.

According to both Company C and D, duplicates amongst requirements are problematic. Often, the same idea or solution comes up several times, but takes different shapes. In addition, these could be mutually exclusive, i.e. they cannot exist at the same time within the system e.g. since they are alternative solutions to the same idea.

4.7 Requirements are invented rather than discovered

Most of the interviewed companies express a trade-off between their own innovative requirements and the requirements suggested by customers and users. At both Company B and C, approximately half of the requirements are suggested by users and half are internally invented. At Company B, both interviewees express a wish to be more market-driven. “That is alpha and omega when creating a product success, to run it from marketing”, as stated by the project manager, continuing “R & D can always create cool things, […] and they think of themselves as ordinary users”. As engineers tend to be early adopters of new technology, developers may not represent the ordinary market for these high technology products.

One of the risks acknowledged by Company B is that when a technology-focused company is founded there is no need to think about who the customers are; it is enough to have some new piece of technology. This is a dangerous situation since, eventually, the company needs to find a customer base and adapt the product to actual customer needs. There is also a need to find the right distributors and retailers. “A product may be incredibly good, but in the wrong store it will still not be sold”, is the comment from the project manager at Company B. The same person also states that in a new technology-focused company, “the developers are the heroes" and therefore marketing has trouble influencing the product functionality. At the same time it is understandable that customer input is difficult to obtain when the product is unique and not known by users. The market simply does not know what to wish for.

At the focus group meeting several participants expressed problems with getting the necessary market input, "[…] those teams consist of happy engineers who invent requirements rather than listen to me" as one of the participants stated. The focus group participants also discussed non-functional requirements (NFR). One participant stated that NFRs often are suggested by customers, while functional requirements often are internal. This could also reflect the fact that many companies are technology-focused and that functional issues therefore are perceived to be more important than non-functional issues.

Although the interviewed organisations are all in the market-driven area, several of them have distributors and retailers who sell the products and it can therefore be difficult to reach end-users for feedback. Thus, both retailers and end-users need to be considered during requirements analysis. In Company A and C, the number of customers is fairly limited, and some large customers have more influence than others. This is also acknowledged by the marketing person at Company E, who describes problems with getting the functionality generic enough to suit several customers.

4.8 Implementing and improving RE within the organisation

A major challenge is to get acceptance for new ways of working with RE, i.e. to implement changes to current practices within an organisation. It includes changing the behaviour of people. Succeeding with the change process is, however, crucial in order to improve RE. A focus group participant stated: "The change process is the most difficult - to change the way people work. There is a lot of good ideas, tools and methods, but it is too tedious to start using them." Both interviewees at Company F state that the biggest challenge is to make everyone follow the same process and work in the same manner.

It is important to realise that it takes time to implement a new way of working, and education of the employees is required. To make a group of developers embrace a certain way of working requires a high level of confidence in the person responsible for the implementation of the new approach. Company F has defined a role called process engineer, who is responsible for implementing process changes and educate employees. The process engineer works together with the project as a support function in matters related to how the process should be carried out. The process engineering also puts effort into pre-phase training of employees in order to improve their ability to understand and use the defined process.

Interviewees from both Company A and B state that the project manager controls how the process is carried out in practice, and therefore the process may vary between projects. It is necessary to obtain acceptance for putting effort into RE amongst project managers before implementing changes in RE practices, i.e. the ideas of improving RE must be well supported by the actors who control the way of working within the organisation. Some participants at the focus group meeting highlighted the issue of transferring knowledge between projects. It is difficult to make people learn from success stories, and apply this knowledge in other projects, especially in organisations where the acceptance for putting effort into RE is low. One focus group participant stated, "All projects where RE has been used, have been successful. However, it is not certain that project managers who have used RE in one project use it in the next one."

4.9 Resource allocation to RE

One major problem, emphasized during the focus group meeting, is the lack of resources to conduct RE. Participants asked for example: "How do you get time and resources to the requirements work?". The problem was explained by a lack of understanding for the time and resources needed to carry out high quality RE, e.g. amongst management. Lack of resources for RE may cause many of the other problems discussed during the meeting, e.g. badly formulated requirements.

Both Company A and B have experienced problems with obtaining enough resources for RE. Often, implementation begins as soon as the project starts, i.e. it is carried out in parallel with the RE work. As a consequence, the ability to put effort into RE is very low, especially since key personnel are caught up in other projects. Interviewees at Company A and B express a need to start and finish the RE work before implementation begins.

Most of the companies estimate that they spend approximately 10-15% of the total effort on RE, although none of the companies actually measure it. However, their opinions regarding whether this is enough or not differ. Company A and B state that too little effort is spent on RE and that much could be improved in their RE practices, while Company C and D consider it to be just the right amount of effort.

4.10 Organisational stability

Software engineering is a knowledge intensive activity, and is naturally dependent on the development staff's knowledge about the product, its usage environment, and their knowledge about the development process. The competence of involved staff is critical when the process is ad hoc. The project manager at Company B said: "Everybody knows everybody, and everybody helps everybody. That is the reason a small company can survive with an ad hoc process" and "The projects depend on the individuals. Some projects would have died if some people would have left". Of course, it is risky to be too dependent on individuals, a problem that became apparent when the company downsized and lost part of its staff. Competence and knowledge was lost. Company C and D also discussed the knowledge capital in the staff and emphasize that their success is thanks to the employees.

Other organisational issues such as co-ordination and communication are key issues within the organisations involved in the study. Company F describes problems with deciding how to organise projects in order to make co-ordination and communication as efficient as possible. Larger projects enhance the overview but complicate co-ordination and they are difficult to manage. Smaller projects facilitate communication within the project group, but make it more difficult to create a comprehensive, overall picture. The systems manager stated, "It is always a matter of how you choose to divide, so to speak. Either you get problems with communication or you get problems with getting the overall picture right".

4.11 Selecting the right process

Several of the interviewed companies struggle with implementing, evolving, or adjusting their development process. Company B is about to develop and implement a development process as the lack of structure in the current way of working is considered to be a major problem. The project manager said: “Yes, I believe that there is an infinite amount of drawbacks with our current way [of working].” The process is a way for project managers to communicate and interact with the different parties involved in a project. Both interviewees at Company B also emphasised that, to start with, only the most important elements should be included in the process. It is easier to evolve the process gradually, rather than trying to do everything at the same time.

Company A, used to had a very detailed process, but it was slimmed down when the product became more mature and fewer changes were made in each release. Unfortunately, it became too slim, which resulted in problems related to lack of structure and control. Therefore, they are still struggling with defining a process on a suitable level of detail.

Company C and D were generally pleased with their way of working. Even though the processes were not rigorous, they had control of their requirements and use tools to structure and maintain the requirements. Both these organisations are fairly small and may therefore have less need for structure and processes.

Company F has put a lot of effort into constructing a specific development process that everyone involved follow. However, they emphasised the importance of making the process flexible enough to be adjustable due to specific situations. Using process engineers is one way of achieving a good interaction between projects and process, and to facilitate the evolution of the process within the projects.

When adjusting the development process, other support processes, such as project management processes need to be adjusted as well. Company F had experienced some problems with combining the iterative RUP process with the sequential Toll Gate (TG) (Cooper, 2001) model demanded by management. Certain business decisions require certain information, which is not produced until later iterations. The systems manager said: “the planning of each iteration cannot be finished at TG2 because then the RE work is not iterative”. In order to ensure that these processes are aligned and support each other, it may be necessary to adjust them, e.g. by redefining the business decisions stipulated by the TG process.

4.12 Release planning based on uncertain estimates

Release planning requires consideration of many different aspects. Most of the companies had some cost-value approach, and consider aspects such as e.g. customer or user value, development cost, strategic value, and marketing positioning in their plan. There are also other aspects, such as effects on the architecture, implementation risks, interdependencies, and deadlines. For embedded systems, even more factors are added, such as component size and availability, as well as sales volumes.

The interviewees at Company F stressed the importance of considering the hardware and non-functional requirements before applications, as architectural aspects may change the structure of the system. These interrelationships are visible in their "anatomy plan", which means that interrelated use cases and functionality can be developed in the same release. The systems manager said: "We want the requirements in one iteration to be connected so that we don't have to change the same documents during the next iteration".

Requirements prioritisation was discussed by all participants in the study. Several companies tended to use some sort of numeral assignment (Karlsson, 1996) to arrange requirements into groups. Requirements are often divided into high, medium and low priority, or given a number between 1 and 5. At Company A, the project manager constructed a top-10 list including the top priority requirements. Another 10 requirements were regarded as medium priority and were put below in the list. Other companies, such as Company C, had a more rigorous approach and used a requirements management tool for prioritisation. At Company B, requirements prioritisation was more ad hoc, or as the project manager said when discussing different stakeholders: "the one who shouts the loudest wins".

Another release planning issue regards effort estimates. Several of the interviewees expressed that it is an important task. Yet, none of the interviewees mentioned any particular method for estimating effort. At Company C the importance of effort estimates during release planning was stressed: "It is important that the requirements end up in the right release and if we estimate incorrectly it will affect the release plan". At Company G, interrelated requirements were sometimes grouped and estimated if they were to be implemented together, because it could be easier than estimating each requirement separately. The opposite opinion was expressed by the project manager at Company D who said that smaller requirements are easier to estimate correctly. Company F usually estimated effort per use case instead of per feature because each feature could affect several use cases. At Company C, the developer said that it is necessary to have detailed knowledge of the system in order to know which parts of the code that is affected by a new requirement.

At Company C and D, effort estimates were discussed between project members so that contradicting estimates were resolved. The developer at Company C said: "we know that we have done it correctly when everyone agrees on the estimate", and at Company D, the project manager said that "we get better estimates if people have to explain how they think". Both Company C and D had weekly meetings, where requirements were discussed, effort estimated and release planned. At Company A, meetings were held with marketing personnel and developers so that those who suggested requirements, i.e. marketing personnel, can explain the purpose to the developers. The developers found it easier to estimate the development effort for requirements when they understood the purpose. The product manager said, "High-level requirements are difficult for developers to estimate".

4.13 Summary of findings

This section summarises and discusses the challenging areas presented above. Table 2 provides a list of the discovered challenges together with a summary of some of the problematic issues, called *key problem areas*.

Although the focus is on market-driven organisations, some of the findings are of more universal relevance. Issues related to communication and coordination may appear also in customer-specific organisations, as well as issues regarding requirements traceability and interdependencies. However, five of the challenges are directly related to the market-driven characteristics of the participating organisations. Those challenges are marked in Table 2 and discussed below.

The special stakeholder characteristics, i.e. several potential customers and users on a large and open market, contribute to some of the challenges. For example, the constant flow of requirements (Section 4.4) is caused by the variety of stakeholders who have demands on the product and like to contribute with their ideas. In relation to the stakeholder characteristics, it can be mentioned that the issue of writing understandable requirements (Section 4.3), and understanding the stated requirements suggestions, is more complex when the stakeholders are

Table 2. Summary of discovered challenges

Challenges	**Key Problem Areas**	**Special for market-driven organisations**	**High importance areas**
4.1 Simple techniques for basic needs	Tailoring of processes, tool integration, complex specification languages, support for customer feedback		*
4.2 Communication gap between marketing staff and developers	Goal alignment, organisational coherence, information flow, notion of "good requirement"	*	
4.3 Writing understandable requirements	Quality of requirements specification, tracing from requirement to origin, capturing rationale	*	*
4.4 Managing the constant flow of new requirements	Trade-off between elaborate elicitation and information screening, efficient database management, feedback on proposals, finding new requirements among bug reports	*	
4.5 Requirements volatility	Trade-off between volatility and stability, early feedback on product		
4.6 Requirements traceability and interdependencies	Dependency impact analysis, requirements bundling structure, NFR impact trade-off, reduction of redundancy		*
4.7 Requirements are invented rather than discovered	Innovation vs. customer needs, balancing different customers' impact on next release	*	
4.8 Implementing and improving RE within the organisation	Continuous improvement, consistent process enactment, change implementation, staff training		
4.9 Resource allocation to RE	Management support, motivating RE resources, resource competition		
4.10 Organisational stability	Reliance on individuals, co-ordination and communication		
4.11 Selecting the right process	Suitable level of detail, basic and minimal RE process to evolve from, combining sequential and iterative development processes		
4.12 Release planning based on uncertain estimates	Managing cost and value drivers, requirements dependencies, requirements prioritisation, effort estimation	*	*

diverse and express their needs vaguely, which is often the case in the market-driven situation. Using natural language, one word may have different meaning to different people. In addition, there is no direct link between the stakeholders and the developers, typically requiring the marketing department to elicit and document requirements. Several of the interviewees requested more communication between the marketing department and developers (Section 4.2). This is especially important in market-driven organisations since the marketing department acts as an interface to the customers and end-users. Although this important interface is the main source of user requirements, some organisations apparently have a more technology-oriented focus where requirements are invented in-house rather than elicited (Section 4.7). Although innovation is necessary in technology-intensive organisations, it may turn focus away from what the users actually need. For example, the focus group participants discussed that users often suggest non-functional requirements, while the functional requirements are in focus internally. Finally, release planning (Section 4.12) is of high importance in the competitive market-driven situation since it is necessary to release products with time-to-market in mind. In order to successfully optimise the user value and development effort for each release, techniques for requirements prioritisation and effort estimation are needed.

The challenges that are special to market-driven organisations are evidently in need of more investigation. However, it is not necessarily those challenges that cause the most problems for the market-driven organisations. The semi-structured type of interviews that were held did not reveal if certain challenges are more important or more acute than others. However, during analysis the *code frequency*, i.e. the number of quotations for each category or code, was examined and it indicates some patterns regarding which challenges that were more discussed than others. The following challenges were much discussed by all participants: Simple techniques for basic needs, Writing understandable requirements, Requirements traceability and interdependencies, and Release planning based on uncertain estimates. Therefore, they may be of common interest to market-driven organisations in general. The same four challenges also received the highest number of quotations in total, confirming that they are of high importance to all of the participating organisations. However, they are not necessarily the most important and cost-effective challenges to deal with in practice. The challenges of high importance are also marked in Table 2. Some of the challenges were discussed more frequently by some participants than others. In the analysis, we attempted to relate those challenges to the organisational characteristics of the particular companies. The purpose was to discover a pattern between the deviation of different answers and the characteristics of the organisation, product or process. However, no clear pattern was visible and therefore no conclusions are drawn based on the relation between organisational characteristics and challenging areas.

5. Discussion

This section is divided in three parts. The first one discusses threats to validity in qualitative designs and the measures taken in the presented study to increase validity. The second part presents our experiences from using a qualitative research approach. Finally, the third part relates our results to the findings in literature.

5.1 Threats to validity in qualitative designs

The quality of a qualitative study relies on the quality of the investigator (Robson, 2002). In order to obtain practice and experience in interviewing, and try out the interview questions, two pilot interviews were conducted early on. Still there is a risk that quotations become out of context when divided into separate coded segments (Coffey and Atkinson, 1996). To deal with this we have used *observer triangulation*, i.e. multiple observers and interviewers (Robson, 2002), so that three different researchers were involved during interviews and analysis. The coded segments could therefore be discussed so that a common understanding was gained. We have also used *data triangulation* (Robson, 2002) shown in the fact that both interviews and focus groups were used as methods for data collection.

We have chosen to divide the validity issues into the three groups described in Robson (2002): *description*, *interpretation* and *theory*. The main threat to providing a valid *description* of what has been seen or heard lies in the inaccuracy or incompleteness of the data. This has been met by audio-taping all interviews and, later on, transcribing them. Furthermore, each interview was carried out by 2-3 researchers who took extensive notes and collected drawings and sketches that were made by the interviewee.

The main threat to providing a valid *interpretation* is that of imposing a framework or meaning on what is happening rather than this emerging from what is learnt during the involvement with the setting. This does not preclude starting with a set of predefined categories, but these categories must be subjected to checking of their appropriateness, with possible modification. It requires that the researchers demonstrate how the interpretation of the end product was reached. In this research, the threat to interpretation was managed by discussing the interviews and how the different researchers interpreted the interviewees' answers. This was accomplished by having multiple interviewers at all occasions and in addition, a fourth researcher read and commented on the transcripts. Furthermore, the transcripts were analysed using the qualitative analysis tool Atlas.ti. In the tool, the codes, or categories, made it possible to trace the route by which we have come to a certain interpretation or conclusion.

The main threat to *theory validity* is in not considering alternative explanations or understandings of the phenomena under study. This can be countered by actively seeking data which are not consonant with the theory. We have accomplished this by seeking new types of organisations for our study, where differences in size, age and business type occur.

Validity can also be discussed in terms of *reliability*. In fixed design research, reliability is associated with the use of standardized research instruments, such as formal tests and scales. In qualitative or flexible designs, formal reliability testing cannot be used. However, there are common pitfalls in data collection and transcription that need to be avoided. Among other things, transcription errors are explained in (Easton et al., 2000). Transcription errors are difficult to avoid due to misinterpretation and mishearing. Inaccurate punctuation or mistyped words can change the entire meaning of a sentence and transcripts do not capture intonation, hesitation, and thought pauses. In this study, the transcriber had also been present at the interview and had heard the answers. In most cases the transcriber was also the one in closest contact with the interviewee and the company, and she/he had thereby a reasonable amount of knowledge about the company culture and language.

Another issue regarding validity is the possibility to generalize the results. *Internal generalisability* is concerned with conclusions drawn within the setting studied. This means that the interviewees or observed situations should not be biased by the researcher. This was managed during sampling as we selected interviewees and organisations from different industrial networks and geographical areas. *External generalisability* concerns conclusions drawn beyond the setting. Qualitative studies rarely attempt to generalise beyond the studied setting, as it is more concerned with characterising, explaining and understanding phenomena under study. Statistical generalisation is often not permitted due to the lack of representative sample. The nature of qualitative designs also makes it impossible to replicate since identical circumstances cannot be re-created. However, the development of a theory can help in understanding other cases and situations. The fact that several of the discovered challenges are acknowledged by all of the participants increases the possibility of transferring the results to other situations. All of the reported challenges are acknowledged by more than two thirds of the participants. Thus, despite the diversity in organisational characteristics, the market-driven type of development seems to create some commonality in challenging areas.

5.2 Experiences from using a qualitative research approach

Qualitative research is rarely used in the software engineering discipline. However, software engineering is in many aspects a social activity and is therefore difficult to understand completely based on quantitative methods. Qualitative methods can give in-depth understanding of social or cultural phenomena. However, some issues make qualitative research difficult in the software engineering discipline. First of all, software is involved in a variety of domains since it is present in various kinds of products. Therefore, the differences in organisational characteristics between the companies involved in the study are large. Thus, it was difficult to know when to stop searching for additional companies to represent the market-driven software development area. The software engineering area is also very fluctuating, which complicates the qualitative research. In our study, it was difficult to extract facts about some of the participating companies since staffing changes frequently, project sizes vary, and documentation is weak. It was also difficult to know whether the process descriptions given by interviewees were examples of how they actually work or how they are supposed to work.

We spent quite a long time adjusting the interview instrument before starting the interviews. While this gave us questions that had been well thought through, spending more time on actually trying the questions out on pilot interviews could have saved more time on the whole. The pilot interviews turned out to be useful, both to adjust the questions and to gain interviewing experience. Later on, multiple interviewers participated during interviews and additional questions were posed to complement the interview instrument. This resulted in a more comprehensive understanding because the different researchers had different focus.

After seven interviews, the data consisted of approximately 100 pages of printed transcripts. This was short enough to manage manually by underlining and marking interesting text segments. However, after the additional seven interviews, the material grew very large. Therefore, it was decided to use a data analysis tool to help us be systematic and rigorous in searching for, and retrieving, data. There is no great conceptual advantage over marking the transcripts physically with code words, but in practice the tool can offer many advantages (Coffey and Atkinson, 1996). It was possible to code at two different sites and then merge the data and the categories together to one single unit. Other benefits include the possibility to search for codes, quotations and words, as well as producing and extracting overlapping codes. These advantages would not have been present with coding on paper. However, one problem was discovered when trying to validate the results. As the results in this paper are written in English and the interviews were carried out in Swedish, the traceability was inferior due to translation problems. To avoid this, memos could have been made in the tool regarding the quotations that were used in the results. In that manner, it would have been easier to search for the memos instead of manually searching through all quotations, which was necessary in our case.

One problem, partly resulting from tool-usage, was that we tended to "over code" the material. The codes, or categories, were too comprehensive so too many quotations could fit into them. And in fear of losing interesting information, sometimes less interesting quotations were marked as well. This resulted in a massive amount of quotations during analysis; for some categories more than 100 quotations were found. A more strict definition for each category, and routines for adding new as well as more delimited categories, could have been helpful to extract the most interesting quotations and less time could have been spent on analysis of the massive material. This was not as problematic during manual coding in the first study, as the amount of quotations was more apparent in the printouts.

Two researchers with different backgrounds cooperated during coding and analysis. It resulted in slightly different findings and gave us the opportunity to double-check and discuss each other's material from two different viewpoints, and thus increase validity. However, as each person had an in-depth knowledge of a subset of the data it was difficult for one person to get a comprehensive view of the whole material. It was also more complex to include new codes or adjust the coding schema.

5.3 Results related to literature findings

Although Curtis et al. (1988) performed their survey several years ago, focusing on large systems development, the study presented here indicates that some problems remain. Communication and co-ordination are still corner stones in software development and project success depends heavily on the skills of the staff involved.

The survey by Hall et al. (2002) also confirms that organisational problems, for example lack of skilled personnel and high staff turnover, have a larger impact than technical problems when it comes to requirements engineering. This is in agreement with our results with regard to organisational stability; downsizing negatively affects organisational knowledge and competence and makes it hard to survive with an ad hoc process. Another problem identified by Hall et al. (2002) is that sometimes the sales staff agrees to deliver unrealistic system features without considering technical and schedule implications. This was also discussed during our focus group as being a challenge.

Lubars et al. (1993) express that the authors were seldom understood when asking the interviewees about which process they follow when writing requirements specifications. This does not seem to be the case in our study as all interviewees had good knowledge about the different steps of their development process. Another difference is that in the paper by Lubars et al. some of the market-driven projects did not produce any written specifications, while this was not discovered in any of the companies we interviewed. Possibly the awareness of the process and the practice to write specifications have been enhanced since the paper by Lubars et al. was published.

The issue of user participation was stressed by El Emam and Madhavji (1995). In one case they expressed that, "[Developers] try to force their ideas on the users". This was also acknowledged in our study, as there is simply no discrete set of users to invite for participation. Instead, there is a tendency in some of the companies towards increased technology focus, as requirements are invented by developers rather than elicited from potential users. Similarly to (Hall et al., 2002) and (Lubars et al., 1993), several of the problems encountered in El Emam and Madhavji (1995) are organisational rather than technical.

Chatzoglou (1997) discusses a lack of resources, i.e. people involved, time and money, for RE activities. Lack of resources was also discussed in our focus group meeting, explained by a lack of understanding amongst managers. Lack of resources was also stated as a reason for badly formulated requirements. Most of the participants we asked, allocate 10-15% of their time for RE. This was found to be enough by Company C and D, while the larger Company A and B find it to be too little.

As a complement to other surveys, the presented study describes a communication gap between marketing and developers, resulting in insufficient effort estimates and requirements quality. The balance between marketing and developers' requirements decisions is also recognized as a dilemma, especially since new requirements arrive constantly. The use of a requirements database rather than a traditional, monolithic requirements specification is also salient, as well as the need to group requirements into bundles to ease requirements structuring and work partitioning.

6. Conclusions

This paper has presented several challenging issues with regard to market-driven RE, found during 14 interviews and a focus group meeting. We have not aimed at generalisable results, as the qualitative research approach is intended to characterise and to find variation rather than similarity. This has affected the design of the study since the participating companies had diverse characteristics (see appendix). Most of the discovered challenges are of an organisational and social nature rather than a technical one. Among the organisational challenges we find issues such as organising projects so that co-ordination and communication is enforced, and obtaining enough resources to carry out RE. Other social issues focus on how to make marketing and development communicate regarding requirements, and how to encourage people to change their way of working, especially when implementing and improving RE in the organisation. Among the technical issues we find issues regarding release planning, techniques for requirements prioritisation and effort estimation, as well as requirements traceability and interdependency problems.

The results of the presented interview study are intended to be useful to researchers in identification of new and industrially relevant research areas. The results can also be useful to practitioners who want to learn from empirically observed obstacles in market-driven software development and base their improvement efforts on such knowledge. In conclusion, the presented results of this study enhance previous industrial surveys on requirements engineering by both corroborating previously found empirical evidence and by adding previously unreported observations to the understanding of industrial requirements engineering.

In Section 4.13 we identified four high importance challenges, which were discussed frequently by all participants: Simple techniques for basic needs, Writing understandable requirements, Requirements traceability and interdependencies, and Release planning based on uncertain estimates. Either one of these four challenges can be used as a starting point for research hypotheses and research agendas because of their high importance to the investigated companies. However, these four challenges might not necessarily be the most important and cost-effective ones to concentrate on in practice. Other research needs to be conducted to find out which challenging areas that need most attention and support.

The presented study can be investigated further by increasing the sample of participants with e.g. large- and medium-sized companies, companies developing embedded products, and companies using an agile development approach. These types of companies are only represented to a small extent in the current study and could therefore widen the scope if included in a future survey. In addition, it would be possible to combine the conducted survey with a questionnaire, since the knowledge gained in this study has provided a foundation for constructing closed questions. Typical questions could regard e.g. which kind of development process that is used, which RE tool that is used, which kind of specification language that is used, how requirements traceability is handled, to what extent requirements are invented in-house as opposed to elicited from the market, which kind of requirements prioritisation technique that is used, which kind of cost estimation technique that is used, etc. A questionnaire of that kind could be sent to numerous companies with a market-driven focus to get a comprehensive picture of the processes, tools and techniques that are used in market-driven industry today.

Appendix

Table A. Summary of the interview instrument

Characterisation	
1.1	Tell us about the company (number of employees, age, business area, etc.)
1.2	Tell us about the company's product/products (time on the market, typical customer/end-user, size of product projects, etc.)
1.3	Tell us about your position in the company (role, daily tasks, responsibility, etc.)
Process issues	
2.1	What is the procedure when developing a product? (kind of process, activities performed, documentation developed, special cases, evaluations performed, etc.)
2.2	What is a "requirement" to you?
2.3	In what way are requirements handled? (requirements process, activities, etc.)
2.4	What challenges do you face when working with requirements? What has been successful regarding requirements engineering?
2.5	How much resources are spent on requirements engineering? Continually or in the beginning? How much time would be optimal?
2.6	What is a "good requirement" to you? And to the company? Is the quality of the requirements assessed? How?
2.7	What kinds of decisions are taken during the development of a product? What kind of support is needed in those decisions?
2.8	Is it possible to make decisions too late? What can be the effect in that case?
2.9	How is it decided what to include in the product? How are the requirements prioritised? What is difficult when deciding what to include in the product?
Artefact issues	
3.1	How are requirements documented? What information and attributes are documented about the requirements?
3.2	What support and what tools do you use to document your requirements? What pros and cons do these tools have?
3.3	How many requirements are handled in a typical project? Who suggests the requirements?
3.4	What kinds of dependencies between the requirements have you come across? Are dependencies documented? Are dependencies actively looked for?
3.5	How do dependencies affect product development? How is it handled?
3.6	Do you group the requirements? How are the requirements groups handled during development?

Table B. Company characteristics

	Company A	**Company B**	**Company C**	**Company D**
Total number of employees	1200	65	13	15
Age	20	5	5	2
Product in focus	Software development tool	Embedded software with focus on image processing. 3 product lines.	Software development support tool focus on requirements management.	Communication tool for distributed groups working with software development.
Customer	Software developing companies, mostly in the telecom industry.	Consumer electronics retailers and distributors.	Software developing companies.	Safety critical software developing companies.
End user	Developers at software developing companies.	Bankers, students, office personnel.	Developers and managers at software developing companies.	Managers and developers at safety critical software developing companies.
Interviewee(s)*	1. Req engineer/ project manager (1) 2. Product manager (2)	1. Project manager (1) 2. Product manager (1)	1. Founder and managing director (1) 2. Head of developers (1)	1. Project manager (1)
Role/responsibility of the interviewee(s)	1. Co-ordinate resources and clarify, define and prioritise requirements. 2. Coordinate between marketing and development, elicit reqs from the market and decide product evolution.	1. Introduce process and break down the requirements specification in several parallel projects. 2. Coordinate between marketing, development and production. Suggest and write requirements.	1. Product management, release planning, marketing, and sales. 2. Regular development tasks, allocate work between developers, and suggest, document and time-estimate requirements.	1. Process responsibility and business analysis, involved in daily development tasks.
Release size and frequency	New release every 6 months to include new requirements and correct detected errors. Involves 30-60 part-time employees in the development of the product in focus.	Includes 25-30 employees for a new product line and 5-15 employees for a new version of existing product. Initiated because of new requirements or when many errors are detected.	New release approximately every 6 months. Each release divided into functions, which involves 1-3 employees for 1-10 weeks.	The product in focus involves 7 employees at 2 locations. At least one release every month.
Process	Defined but not documented, followed thanks to the experienced staff.	Introducing process. The ad hoc process works thanks to the experienced staff.	Elaborate. Documented and integrated in the tool they develop and use. Based on requirements status. Incremental development within each release.	Elaborate. Extreme Programming, pair programming excluded. Incremental development within each release.
Requirements documentation	Natural language. Support system database where customers and developers add requirements. A web site with the ones for the current release written on a high abstraction level.	Natural language, state charts and UML in the requirements specification. Changes to the specification are rarely documented.	Natural language. Database included in the tool where customers and employees can add requirements.	Natural language. Customers' requirements suggestions on virtual story cards, which can be followed through the process.
Project or process evaluation	Not regularly. Have downsized the process but improved requirements awareness.	Process under construction. Have increased overall system and product awareness, e.g. within requirements and testing. Introducing project evaluations.	Included in the process. Evaluation occurs after each release and has e.g. adjusted the status range.	Weekly meetings to sum up the week's problems. Individual time surveillance.

Table B. Company characteristics

	Company E	Company F	Company G	Company H
Total number of employees	12	460	300-400	26
Age	1	2.5	18	16
Product in focus	Software development and visualization tool for integration of electronic business processes.	Switch for supporting mobile telecommunication.	Network communication solutions for printers.	Software tool to support sales configuration for large products.
Customer	Large companies that need improved business processes.	Telecommunication operators.	Systems administrators at companies that use printers.	Companies with large configurable products.
End user	Developers of electronic business processes.	Users of mobile phone technology.	Printer users.	Sales personnel.
Interviewee(s)*	1. Founder and managing director (1) 2. Product manager (2)	1. System manager (2) 2. Process engineer (2)	1. Software developer (2)	1. Customer relations (2) 2. Project manager (2)
Role/responsibility of the interviewee(s)	1. Marketing, customer contacts, and release planning. 2. Product and technology responsibility. Head of development, control the overall development process.	1. Coordinate between product management and development. Refine the high level reqs into detailed reqs. 2. Define and improve the RE process. Process introduction. Support the projects with method/ process related issues.	1. Regular development tasks; participate in analysis, design and implement requirements (coding), etc.	1. Customer responsibility for their largest customer. Elicit and analyse requirements. 2. Regular development tasks, such as analysis and design, coding, etc. Control and plan the project.
Release size and frequency	No regular releases, puts together a project including all employees when many bugs or requirements have been identified.	Regular releases, approximately once a year (earlier every 6th month). Involves over 100 developers, divided into teams.	No regular releases, puts together a project on demand. Involves 4-8 persons and last about 6 months.	Aims at releasing two new versions per year. Involves 6-7 persons.
Process	Not documented, but the experienced staff knows the activities by heart.	Elaborate and defined. A RUP process especially adjusted towards their needs. Incremental combined with tollgate-based process.	Defined and documented. Have a development manual.	Defined with documentation templates.
Requirements documentation	Natural language in the requirements specifications and UML at a later stage.	Natural language and UML, especially in later stages. Several documents, for various levels of detail e.g. one for Use Cases.	Natural language. Have two documents for requirements on different levels of detail.	Natural language together with use cases, and other UML diagrams when needed. All reqs are documented in Excel, and for each selected reqs a functional specification is made.
Project or process evaluation	Not regularly but improvements ideas are discussed.	Regular evaluations. Have staff especially responsible for improving the process (interviewee 2).	Not regularly. The project manager adapts the process to the project-specific needs.	Not regularly, as part of the development process, but lots of improvements on the way.

*The interviewees are marked with a (1) if participating in the first part of the study and a (2) if participating in the second part of the study

PART II: EVALUATION OF REQUIREMENTS PRIORITISATION TECHNIQUES

Paper 2. Pair-wise Comparisons versus Planning Game Partitioning - Experiments on Requirements Prioritisation Techniques.
Lena Karlsson, Thomas Thelin, Björn Regnell, Patrik Berander, Claes Wohlin
Accepted for publication in Empirical Software Engineering Journal, 2006.

Paper 3. Evaluating the Practical Use of Different Measurement Scales in Requirements Prioritisation.
Lena Karlsson, Martin Höst, Björn Regnell
Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE'06), Rio de Janeiro, Brazil, September 2006.

Abstract

The process of selecting the right set of requirements for a product release is dependent on how well the organisation succeeds in prioritising the requirements candidates. This part of the thesis is based on two papers regarding experimentation on requirements prioritisation techniques.

The first paper describes two consecutive controlled experiments comparing different requirements prioritisation techniques with the objective of understanding differences in time-consumption, ease of use and accuracy. Among the three investigated techniques, the Tool-supported pair-wise comparisons is the fastest, and the Planning game is the second fastest technique. The techniques do not differ regarding ease of use. The manual Pair-wise comparison technique is the most time-consuming and least easy to use. The techniques do not differ significantly regarding accuracy.

The second paper presents an empirical investigation of differences between the measurement scales used in requirements prioritisation. The ratio scale is richer than the ordinal scale as it provides more information, but ratio scale techniques are often more complex to use. A measure is presented, describing the characteristics of the ratio scale prioritisation results. It can be used to compare results from different prioritisation sessions. In addition, possibilities for using the cost-value approach with ordinal scale data are evaluated with promising results.

1. Introduction

In market-driven software development, products are developed in several consecutive releases intended for an open market. When requirements are elicited from several stakeholders on an open market, it often yields more requirements than can be implemented at once. The requirements need to be prioritised so that the most significant ones are met by the earliest product releases (Wiegers, 1999; Siddiqi and Shekaran, 1996).

During a project, decision-makers in software development need to make many different decisions regarding the release plan. Issues such as available resources, milestones, conflicting stakeholder views, available market opportunities, risks, product strategies, and costs need to be taken into consideration when planning future releases. In particular, the cost-value approach takes both development cost and customer value into account. As software development companies have limited resources, it is essential to choose the requirements that give the best return on investment, in terms of customer satisfaction. Unfortunately, there is a lack of simple and effective techniques for requirements prioritisation, which could be used for release planning. If an organisation fails to determine the most important requirements, it risks that the developed system does not meet customers' needs and expectations (Karlsson and Ryan, 1997).

The software literature includes many sources that state the importance of prioritising requirements. In the field study by Lubars et al. (1993), several companies expressed a need for guidance in assigning, modifying and communicating requirements priorities. Lehtola and Kauppinen (2004) states that in many companies requirements prioritisation practices are still mostly informal. Siddiqi and Shekaran (1996) identified requirements prioritisation as an important, though disregarded, issue in RE research at that point in time. It seems as requirements prioritisation still need further attention in research.

There are several different techniques for requirements prioritisation. Some techniques result in priorities on an ordinal scale, and provide the ranked order among requirements, e.g. the Numeral assignment (Karlsson, 1996) and the Planning game (Beck, 2005). Other techniques provide the result on a ratio scale, and state how much more important one requirement is than another. Examples of these techniques are the Pair-wise comparisons (Saaty, 1980), Wiegers' method (Wiegers, 1999), and the $100 test (Leffingwell and Widrig, 2000).

Scales that contain more information than others are called richer (Fenton and Pfleeger, 1997). Hence, the ratio scale is richer than the ordinal scale as it provides the relative distance between ordered requirements in addition to the ranks. Techniques providing the result on a ratio scale are often more time-consuming and complex to use than techniques based on an ordinal scale (Lehtola and Kauppinen, 2004). Therefore it is interesting to investigate whether or not the added information is valuable to the decision-maker.

Part II aims at investigating differences between different requirements prioritisation techniques and different scales used in requirements prioritisation. This is done in two studies: one describing two experiments comparing three prioritisation techniques[3], and one using archive analysis to investigate results from prioritisation sessions. The two studies investigates different research questions and have different goals and therefore the results are discussed and concluded in separate sections, while the introduction and related work is described in joint sections.

The first study aims at comparing three different requirements prioritisation techniques: Tool-supported pair-wise comparisons, Planning game, and manual Pair-wise comparisons. Ratio scale techniques, such as the Pair-wise comparisons, are more complex to use than ordinal scale techniques, such as the Planning game. Therefore it would be easier and more efficient to use prioritisation techniques based on the ordinal scale, however, they might not be sufficient as a basis for decision-making. The second study investigates the difference in decision-support between ordinal scale techniques and ratio scale techniques. The study suggests an approach to measure the skewness of the ratio scale distribution and a way to use the cost-value approach on ordinal scale data. Both studies are based on empirical data.

Part II is divided as follows. This introduction is followed by Section 2, describing related work on requirements prioritisation, including different techniques, different measurement scales, and the cost-value approach. Thereafter, the two studies (Paper 2 and 3) follow in Section 3 and 4. Finally, Section 5 provides some closing remarks on requirements prioritisation based on the findings in the two studies.

[3] Information and design is available at http://serg.telecom.lth.se/research/packages/ReqPrio

2. Related Work

This section describes some related work in the areas of requirements prioritisation techniques, measurement scales for requirements prioritisation, and the cost-value approach.

2.1 Requirements Prioritisation

There are several different techniques to choose from when prioritising requirements. Some are based on determining the *absolute* importance of the candidate requirements, by e.g., assigning each requirement a certain priority such as essential, conditional or optional (IEEE, 1998). Other techniques are *relative* and require a person to determine which requirement is more important. Thereby, all requirements get different priorities, whereas absolute techniques may assign several requirements to the same priority. Relative approaches tend to be more accurate and informative than absolute ones (Karlsson, 1996). One relative technique is the $100-test (Leffingwell and Widrig 2000) and another one is Pair-wise comparisons (Karlsson and Ryan, 1997), see below. In addition, there are several techniques aimed at release planning, in particular when several stakeholders are involved, such as EVOLVE (Greer and Ruhe, 2004) and Quantitative WinWin (Ruhe et al., 2002). Both techniques are aimed at release planning of incremental software development. A selection of techniques for requirements prioritisation is described below. For a thorough review of these and other prioritisation techniques, see Berander and Andrews (2005), Lehtola and Kauppinen (2004) and Moisiadis (2002).

2.1.1 Planning Game (PG)

PG is used in planning and deciding what to develop in an Extreme Programming (XP) project. In PG, requirements (written on so called story cards) are elicited from the customer. When the requirements have been elicited, they are prioritised by the customer into three different piles: (1) those without which the system will not function, (2) those that are less essential but provide significant business value, and (3) those that would be nice to have (Beck, 2005).

At the same time, the developers estimate the time required to implement each requirement and, furthermore, sort the requirements by risk into three piles: (1) those that they can estimate precisely, (2) those that they can estimate reasonably well, and (3) those that they cannot estimate at all.

Based on the time estimates, or by choosing the cards and then calculating the release date, the customers prioritise the requirements within the piles and then decide which requirements that should be planned for the next release (Newkirk and Martin, 2001). Thus, the technique uses a sorting algorithm, similar to numeral assignment (Karlsson, 1996), to partition the requirements into one of three piles. Then, the requirements within each pile are compared to each other in order to achieve a sorted list.

The result of the PG technique is an ordered list of requirements. This means that the requirements are represented as a ranking on an *ordinal scale*, without any information about how much more important one requirement is than another.

In the investigation performed by Karlsson et al. (1998) a similar technique, called Priority groups, was investigated. In the Priority groups technique, requirements are put into one of three groups, corresponding to high, medium and low priority. In groups with more than one requirement, three new subgroups are created until no group has more than one requirement. Thereby an ordered list of requirements is compiled. Priority groups was given the lowest subjective ranking (regarding ease of use, reliability and fault tolerance) of the six investigated prioritisation techniques in Karlsson et al. (1998). The technique was ranked as 4th of the six techniques regarding the objective measure total time-consumption.

2.1.2 Pair-Wise Comparisons (PWC)

Pair-wise comparisons involves comparing all possible pairs of requirements, in order to determine which of the two requirements is of higher priority, and to what extent (Karlsson and Ryan, 1997). If there are *n* requirements to prioritise, the total number of comparisons to perform is *n(n-1)/2*. For each requirement pair the decision-maker estimates the relation between the requirements on the scale {9, 7, 5, 3, 1} where 1 represent equal importance and 9 represent one requirement being much more important than the other.

This relation results in a dramatically increasing number of comparisons as the number of requirements increases. However, due to redundancy of the pair-wise comparisons, PWC is rather insensitive to judgement errors. Furthermore, PWC includes a *consistency check* where judgement errors can be identified and a *consistency ratio* can be calculated.

PWC is used in the Analytic Hierarchy Process (AHP) (Saaty, 1980). In AHP it is possible to take the system perspective into account, so that a system structure of related requirements can be abstracted into a hierarchy that describes requirements on different abstraction levels. Hence, AHP can take the whole system into account during decision-making since it prioritises the requirements on each level in the hierarchy (Saaty, 1980).

In the investigation by Karlsson et al. (1998), the authors conclude that PWC (there called AHP) was the most promising approach because they found it trustworthy and fault tolerant. It also includes a consistency check and it is based on a *ratio scale*, i.e., it includes the priority distance. PWC was the only technique in the evaluation that satisfied all these criteria. However, because of the rigour of the technique, it was also the most time-consuming in the investigation.

In another empirical investigation of prioritisation techniques performed by Lehtola and Kauppinen (2004), PWC was compared to Wiegers' method (Wiegers, 1999). The authors conclude that "users found it difficult to estimate how much more valuable one requirement is than another" and that some users found pair-wise comparisons pointless as they felt it would have been easier for them to just select the most important requirements (Lehtola and Kauppinen, 2004).

2.1.3 Tool-Supported PWC (TPWC)

Since the major disadvantage of PWC is the time-consumption for large problems, different investigations have been performed in order to decrease the number of comparisons, and thus the time needed (Carmone et al., 1997; Harker, 1987; Karlsson et al., 1997; Shen et al., 1992). The results of these have been that it is possible to reduce the number of comparisons with as much as 75%. Techniques for reducing the number of comparisons are called Incomplete Pair-wise Comparisons (IPC). The techniques are based on providing *stopping rules*, indicating when additional pair-wise comparisons are no longer necessary (Karlsson et al., 1997). However, when reducing the number of comparisons, the number of redundant comparisons is also reduced. Thereby, the sensitivity for judgemental errors increases (Karlsson et al., 1998).

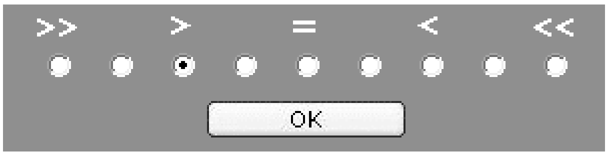


Figure 1. *Part of the user interface in the tool used for TPWC*

The PWC technique described in Section 2.1.2 has been built into a requirements management (RM) tool called Focal Point (Telelogic, 2006). The tool guides the user to apply pair-wise comparisons between requirements in a similar manner as the PWC technique. The tool contains an IPC algorithm and stopping rules that indicate to the user when the necessary number of comparisons has been performed. The number of required comparisons is reduced to the approximate size *2n*, where n is the number of requirements. Thereby, the time-consumption is reduced radically in comparison with the manual PWC.

The tool displays one requirement pair at the time to the user, possibly including descriptions of the requirements. The prioritisation is based on a ratio scale, and applies pair-wise comparisons between requirements based on some criteria chosen by the user beforehand. The user selects one of the nine possible "more than", "equal" or "less than" symbols between the two requirements, as illustrated in Figure 1. When the user clicks "ok", the next pair of requirements is displayed. In that manner the focus is retained, since only one task at the time is presented to the user. As the redundancy is reduced by the IPC algorithm, it affects the quality of the results. The tool includes a consistency check that identifies inconsistencies among the requirement priorities. The user may then revise the inconsistent comparisons until an acceptable consistency is achieved.

The tool also incorporates solutions for RM and project portfolio management and can visualise the prioritisation results in various charts and diagrams.

2.1.4 Numeral Assignment

The Numeral assignment technique is based on the principle that each requirement is assigned a symbol representing the requirement's perceived importance. Several variants based on the Numeral assignment technique exist, e.g. classifying requirements as mandatory, desirable or inessential (Karlsson, 1996). Another way to classify requirements is to divide them into essential, conditional or optional requirements, as suggested by the IEEE (1998). Furthermore, it would be possible to give each requirement a number e.g. between 1 and 5, where requirements with a 5 are the most important ones (Karlsson, 1996). Classifying requirements according to Numeral assignment does not give us information about the relation between the requirements in each class, thus several requirements may appear equally valuable.

2.1.5 $100 Test

In the $100 test, each participant is given $100 in fictional money to distribute between requirements. Each participant is asked to write down on a sheet how much of this money is to be spent on each requirement. Then a facilitator tabulates the results and provides an ordered ranking of requirements (Leffingwell and Widrig, 2000). The total amount of money spent on each requirement provides us with a relative difference between the different requirements, i.e. the results are obtained on a ratio scale. The technique is particularly useful for calculating a cumulative vote based on several participants' views.

2.1.6 Wiegers' Method

According to Wiegers' method, the priority of a requirement can be calculated by dividing the value of a requirement with the sum of costs and technical risks associated with implementing it (Wiegers, 1999). Typical participants are the project manager, key customer representatives and development representatives. The resulting priorities are on a ratio scale.

2.2 Theory of Measurement Scales

In the 50s S.S. Stevens proposed properties of measurement systems and described four different scale types: Nominal, Ordinal, Interval and Ratio, each of which possesses different properties of measurement systems (Fenton and Pfleeger, 1997). The scale types are presented in order of richness, i.e. the second one is said to be richer than the first one as all relations in the second one are contained in the first (Fenton and Pfleeger, 1997). This section describes these four different scale types in more detail.

2.2.1 Nominal Scale

The nominal scale is the most primitive of the four scale types and includes some kind of categorisation or classification. All objects are grouped into subgroups and each subgroup is assigned a certain name or number. No object is allowed to belong to more than one subgroup and there is no ordering among the classes and no notion of magnitude associated with the numbers or symbols (Fenton and Pfleeger, 1997). Requirements grouped according to which sub systems they concern is an example of nominal classification.

The only statistics to be gathered on this scale is frequency, i.e. the number of objects in each group. The mode can be calculated, but not the median or mean.

2.2.2 Ordinal Scale

The ordinal scale can be used to enhance the nominal scale with information about the ordering of classes or categories. This is the case in Numeral assignment, when each requirement is classified according to its value and assigned to e.g. the mandatory, desirable, or inessential (Karlsson, 1996) group. Priorities can also be measured using numbers such as 1, 2, 3, where the requirements with highest priority are assigned a 1. In addition, requirements within the groups can be ranked so that an ordered list of requirements is received. This scheme is used in the Planning game.

The numbers associated with the requirements represent ranking only, so arithmetic operations, such as addition and multiplication, have no meaning (Fenton and Pfleeger, 1997). Statistics to be used on ordinal scales are calculation of the median and non-parametric statistics.

2.2.3 Interval Scale

This scale type carries information about the size of the intervals between the ordered classes, so that we can in some sense understand the jump from one class to another. An interval scale preserves order, as with an ordinal scale, and differences – but not ratios. The interval scale does not have any apparent application in requirements engineering.

2.2.4 Ratio Scale

The richest of the four scale types is the ratio scale, as it possesses ordering, size of intervals and ratios between entities. There is a zero element, representing a total lack of the attribute and measurement start at zero. This scale type is used in e.g. the Pair-wise comparisons. The ratio scale provides not only ordering of requirements, but also the relative distance between ordered requirements, and states how much more important one requirement is than another. All arithmetic can be applied to classes on this scale. Both parametric and non-parametric statistics can be performed on ratio scale data, and the mean can be calculated.

2.3 Cost-Value Approach

When prioritising requirements, it is often not enough to prioritise only how much value the requirement has to the customers. Often other factors such as risk, time, cost and requirements interdependencies should be considered before deciding if a requirement should be implemented directly, later, or not at all. For example, if a high-priority requirement would cost a fortune, it might not be as important for the customer as the customer first thought (Lauesen, 2002). This means that it is important to find those requirements that provide much value for the customers at the same time as they cost as little as possible to develop.

Wiegers (1999) suggests that the value of a requirement is balanced against not just its cost, but also any implications it has for the architectural foundation and future evolution of the product. He also proposes that the value is seen as being dependent both on the value it provides to the user and the penalty incurred if the requirement is absent.

Karlsson and Ryan (1997) use PWC as an approach for prioritising regarding both value and cost in order to implement those requirements that give most value for the money. The cost-value approach can visualise the value-to-cost ratio between the requirements in a cost-value diagram.

The cost-value diagram is used to determine which requirements have a high value-to-cost ratio and which do not (Karlsson and Ryan, 1997). When using the ratio scale there are different ways to determine from the cost-value diagram the requirements with high contribution to the product, i.e. with a high value-to-cost ratio. Similarly, it is possible to find the requirements with a low contribution to the product, i.e. with a low value-to-cost ratio. The cost-value diagram is often divided into three separate areas, marked A, B, and C, below. The two main options to determine these areas are shown in Figure 2. Option (a) is to draw lines so that one third of the requirements end up in each area (A, B and C) and option (b) is to draw lines so that requirements with a value-to-cost ratio higher than 2 end up in area A, and the ones with a value-to-cost ratio lower than 0.5 end up in area C. Then area B will include the requirements in between. Option (a) is utilised in the commercial tool (Focal Point) used in the experiment assignments described below, and option (b) is presented in e.g. (Karlsson and Ryan, 1997). In both cases, the lines start in the origin of the diagram and are drawn diagonally through the diagram.

Requirements in area A are high contributors and should be implemented as soon as possible, as they are valuable but not expensive to implement. Requirements in area C are low contributors and too expensive to implement regarding their low customer value. Requirements in area B are medium contributors and have to be analysed further. When using the ordinal scale, the described procedures are not natural. Drawing lines from the origin of the diagram is not applicable, as the zero does not have any meaning in the ordinal scale, i.e. no requirement can be ranked as zero. As stated before, it is not valid to use arithmetic such as division and multiplication, and therefore the value-to-cost ratio is not feasible when using ranks.

A more feasible option for the ordinal scale would be to divide the graph into a number of squares by drawing vertical and horizontal lines through the graph. Since common techniques such as the Planning game and Numeral assignment involve dividing the requirements into three groups for each criterion, this could also be applied in the cost-value diagram. It would result in nine equally large squares based on the ranks,

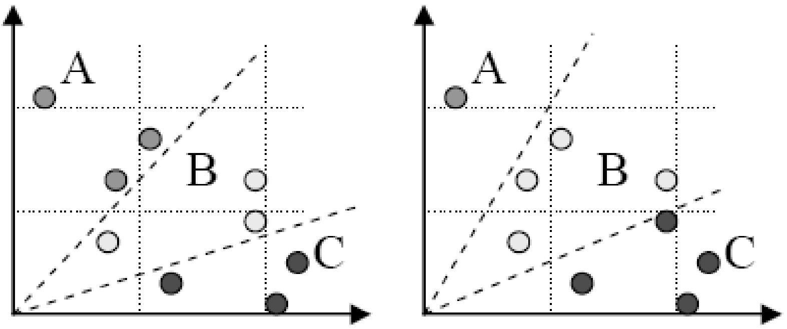


Figure 2. *Option (a): One third of the requirements in each area*

Option (b): Lines are drawn regarding the value-to-cost ratio as shown in Figure 3. We suggest that the requirements in areas denoted A are the high contributors that should be implemented first due to their high value and low cost. The requirements in areas denoted C are low contributors and should be implemented last or perhaps not at all. Requirements in areas B are medium contributors and need further investigation.

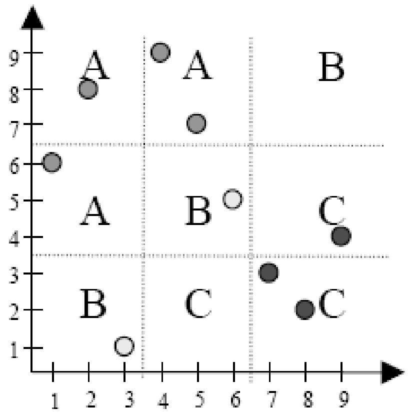


Figure 3. *Ordinal scale cost-value diagram*

3. Pair-wise Comparisons versus Planning Game Partitioning - Experiments on Requirements Prioritisation Techniques

This section presents the two experiments described in Paper 2. The first of the two experiments is also presented in Paper 13.

3.1 Background

Our goal is to *analyse and compare requirements prioritisation techniques for the purpose of gaining increased understanding of the techniques with respect to their time-consumption, ease of use, and accuracy from the point of view of the decision-maker.* The study describes two consecutive experiments aimed at comparing the three requirements prioritisation techniques Planning game, Pair-wise comparisons, and Tool-supported pair-wise comparisons, see Table 1.

The section is structured as follows. Section 3.2 describes the first of the two experiments, which compares a rudimentary prioritisation technique (Planning game) with a more elaborate one (Pair-wise comparisons). As the Pair-wise comparisons turned out to be very time-consuming, a majority of the subjects found it less easy to use and most subjects even found it less accurate, the second experiment was designed to investigate if the technique would benefit from tool-support. In the second experiment, prioritisation with a commercial RM tool (Focal Point) was compared to prioritisation with the manual Planning game, which is described in Section 3.3. The results from the second experiment indicate that the Tool-supported pair-wise comparisons is a faster technique than the Planning game while the ease of use and accuracy are

Table 1. Details about the three techniques compared in the experiments

Technique	**Abbreviation**	**Prioritisation algorithm**
Pair-wise comparisons	PWC	Exhaustive pair-wise comparisons between requirements
Planning game	PG	Sorting algorithm to partition and rank requirements
Tool-supported pair-wise comparisons	TPWC	Tool-support for PWC, reduced number of comparisons

equally high. Section 3.4 discusses the results and compares the two experiments. Finally, Section 3.5 includes some conclusions from the experiments.

3.2 Experiment 1

This section describes the first of the two experiments, the experiment planning and operation as well as the analysis. Finally, it is concluded with a discussion of the results.

The motivation for the experiment is that although requirements prioritisation is recognised as an important area, few research papers aim at finding superior prioritisation techniques that are accurate and usable. This experiment aims at comparing two of the available techniques in order to understand their differences. The PWC was pointed out as a superior technique in a comparison between prioritisation techniques (Karlsson et al., 1998), while a technique similar to PG was ranked rather low. However, the PG technique is of current interest since it is used in the agile community. Therefore these two techniques are interesting to investigate.

The experiment design described in this section is to a large extent also used in the second experiment. Therefore, Section 3.3 is focused on describing the second experiment and the differences between the two experiment designs.

3.2.1 Hypotheses and Variables

The goal of the experiment is to compare two prioritisation techniques and to investigate the following null hypotheses:

$H_0 1$: The average time to conclude the prioritisations is equal for both techniques, PG and PWC.

$H_0 2$: The ease of use is equal for both techniques, PG and PWC.

$H_0 3$*:* The accuracy is equal for both techniques, PG and PWC.

The alternative hypotheses are formulated below:

$H_A 1$*:* The average time to conclude the prioritisations is not equal for both techniques, PG and PWC.

$H_A 2$*:* The ease of use is not equal for both techniques, PG and PWC.

$H_A 3$*:* The accuracy is not equal for both techniques, PG and PWC.

The independent variables are the techniques PG and PWC. The objective dependent variable *average time to conclude the prioritisations* was captured by each subject by noting their start and stop time for each task. The subjective dependent variable *ease of use* was measured by a questionnaire, which was filled out by all subjects after the experiment. The subjects were asked “Which technique did you find easiest to use?” The subjective dependent variable *accuracy* was measured by conducting a post-test a few weeks after the experiment. Each subject was sent four personal lists (two for each criterion), corresponding to the priority order compiled from the two techniques investigated during the experiment. The subjects were asked to mark the priority order that corresponded best to their views. The time-consumption and ease of use are very important measures since resources are limited and a fast and easy technique is more likely to be used than a more effort-demanding one. The third and probably most important variable is the accuracy, i.e., that the technique is trustworthy and that the resulting priority order reflects the decision-maker’s opinion. In a recent case study investigating prioritisation techniques, participants found the resulting priority order incorrect when using Wiegers’ method. Some participants changed their estimates in order to get a better priority order, when the results given by the method seemed wrong (Lehtola and Kauppinen, 2004). This accuracy of the resulting priority order is interesting to investigate and therefore we compare the subjective accuracy of the techniques in this experiment.

3.2.2 Experiment Design

The experiment was carried out with a *repeated measures design, using counter-balancing* i.e., all subjects used both techniques (Robson, 2002; Wohlin et al., 2000). The 16 subjects in the convenient sample included 15 Ph.D. students (10 male and 5 female) in their first or second year, and one professor (male). The experiment was conducted as part of a research methodology course. Before the experiment, a pre-test was performed. The experiment was carried out during a one-day session, which included an introduction to the task, the experiment itself, a post-test, and finally a concluding discussion of the experiment implementation. In addition, a few weeks after the experiment a second post-test was conducted. Figure 4 outlines the activities performed in Experiment 1.

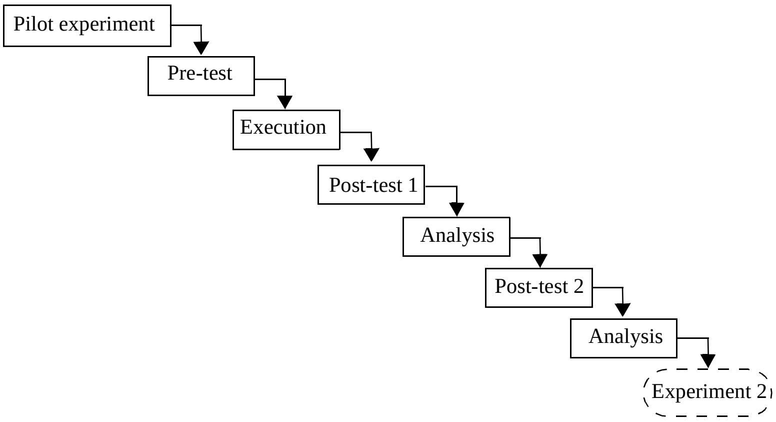


Figure 4. *Activities conducted in Experiment 1*

The requirements used in the prioritisation were mobile phone features, which are requirements on a high level of abstraction and rather independent. The prioritisation was performed without taking requirements dependencies into account.

The trade-off between cost and value, often faced by a development organisation, was difficult to investigate for our subjects, as the cost of developing a certain requirement is difficult for laymen to estimate. Therefore the criterion Price was selected instead, as the trade-off faced by consumers regards the Value of different functions in the phone and the Price of the phone. The criteria are defined as follows:

- The Value criterion corresponds to how important and valuable the subject find the requirement.
- The Price criterion corresponds to how much the subject thinks the requirement adds to the price of the mobile phone.

The Value criterion has probably been regarded by most subjects when buying or comparing mobile phones. The Price criterion may also be accounted for since buying or comparing mobile phones gives a clue of how the price differs depending on the included requirements. Thus, there is a trade-off between Value and Price when buying a mobile phone.

The two requirements prioritisation techniques described in Section 2.1.1 and 2.1.2 were used as input to the experiment, but were modified in order to be more comparable. The PWC is conducted using the AHP for calculating requirements priorities. A flat requirements structure was used, i.e., the system aspect of AHP was not considered in our PWC technique (Saaty, 1980). Neither did we use any of the possible ways of reducing the number of comparisons, thus the pair-wise comparisons were exhaustive. PG was modified so that the piles were labelled according to the Value and Price criteria: (1) Necessary, (2) Adds to the value and (3) Unnecessary, and (1) Very high price, (2) Reasonable price and (3) Low price, respectively. Thus, the aspects of implementation cost and risk, which are emphasised in XP were substituted by Price in our experiment to make it reasonable for laymen to estimate.

Pilot Experiment. A pilot experiment was performed before the main study to evaluate the design. Six colleagues participated and they prioritised ten requirements each, with both techniques. After this pilot experiment, it was concluded that the experiment should be extended to 8 and 16 requirements in order to capture the difference depending on the number of factors to prioritise. Another change was to let the subjects use the techniques and criteria in different orders to eliminate order effects. Further, changes to the PWC sheets included to remove the scale and instead use "more than" and "less than" signs so that the participants would not focus on the numbers, and to arrange the pairs randomly on each sheet.

Pre-Test. Before the session, the subjects were exposed to a *pre-test* in order to get a foundation for sampling. A questionnaire was sent out by e-mail in order to capture the knowledge about mobile phones and the subjects' knowledge and opinions of the two prioritisation techniques. The pre-test was used to divide the subjects into groups with as similar characteristics as possible.

Another objective with the pre-test was to investigate how well the subjects could apprehend the price of mobile phone requirements. A majority of the subjects stated that they consider buying a new mobile phone at least every second year, and therefore we believe that their knowledge of mobile phone prices is fairly good.

Execution. The experiment took place in an ordinary lecture room during a one-day session. Data were mainly collected through questionnaires where the subjects filled out the time spent on each task and their opinions on the techniques.

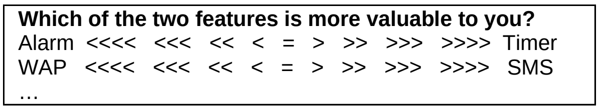
The domain in this experiment was mobile phones and according to the pre-test, all subjects were familiar with this context. The factors to prioritise were mobile phone requirements, for example SMS, Games, WAP, Calendar, etc. (see the appendix, Table C, for complete list).

One intention of the experiment was to investigate if a different number of requirements would affect the choice of preferred technique. Therefore, half of the subjects were asked to prioritise 8 requirements, while the other half prioritised 16 requirements. Another intention was to investigate if the order in which the techniques were used would affect the choice of preferred technique. Therefore, half of the subjects started with PWC and half started with PG. The order of the Value and Price criteria was also distributed within the groups in order to eliminate order effects. Thus, the experiment was performed using a counter-balancing design, as shown in the appendix, Table A.

The experiment was conducted in a classroom with the subjects spread out. Each subject was given an experiment kit consisting of the PWC sheets and the PG cards.

For PWC, one sheet per criterion and person had been prepared, with all possible pair-wise combinations of the requirements to compare. For the purpose of eliminating order effects, the order of the pairs was randomly distributed so every subject received a different order of the comparisons. With 16 requirements to compare, there was 16(16-1)/2 = 120 pair-wise comparisons for Value and Price, respectively. With 8 requirements, there was 8(8-1)/2 = 28 pair-wise comparisons for Value and Price, respectively. In between each pair in the sheets there was a scale where the difference of the requirements' Value or Price was circled, see Figure 5. To be able to try different scales, no scale numbers were written on the sheets. Instead, a scale with 9 different "more than", "equal" and "less than" symbols was used. The further to the left a symbol was circled, the more valuable (or expensive) was the left requirement than the right one. If the requirements were regarded equally valuable (or expensive) the "equal" symbol was circled.

For PG, the subjects were given two sets of cards (one set for Value and one for Price) with one mobile phone requirement written on each. The cards were partitioned into three piles, separately for the Value criterion



Which of the two features is more valuable to you?
Alarm <<<< <<< << < = > >> >>> >>>> Timer
WAP <<<< <<< << < = > >> >>> >>>> SMS
…

Figure 5. *Example of PWC sheet*

and the Price criterion, see Figure 6. The piles represent (1) Necessary, (2) Adds to the value and (3) Unnecessary, for the Value criterion, and (1) Very high price, (2) Reasonable price and (3) Low price, for the Price criterion.

Within the piles, the cards were then arranged so that the most valuable (or expensive) one was at the top of the pile and the less valuable (or expensive) were put underneath. Then the three piles were put together and numbered from 1 to 8 and 1 to 16 so that a single list of prioritised requirements was constructed for each criterion.

The subjects were given approximately 2 hours to conclude the tasks, which was enough time to avoid time-pressure. During the experiment, the subjects were instructed to note the time-consumption for each prioritisation. Further, the subjects had the possibility to ask questions for clarification.

Post-Test 1. The subjects handed in their experiment kit after finishing the tasks and were then asked to fill out a post-test. This was made in order to capture the subjects' opinions right after the experiment. The test included the questions below, as well as some optional questions capturing opinions about the techniques and the experiment as a whole. The questions were answered by circling one of the symbols "more than", "equal" or "less than".

1. Which technique did you find easiest to use?
2. Which technique do you think gives the most accurate result?

Post-Test 2. After completing the analysis, the subjects were, in a second post-test, asked to state which technique that, in their opinion, gave the most accurate result. They were sent two sheets (one for Value and one for

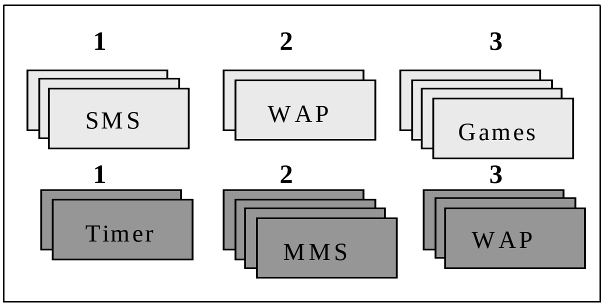


Figure 6. *Example of PG cards*

Price) with two different lists of requirements, corresponding to the results from the PG and PWC prioritisations. The post-test was designed as a *blind-test*, thus the subjects did not know which list corresponded to which technique, but were asked to select the list they felt reflected their opinions the most. In order to get comparable lists, the ratio scale from PWC was not shown, and neither was the pile distribution from PG.

3.2.3 Threats to Validity

In this section, the threats to validity in the experiment are analysed. The validity areas considered are conclusion, internal, construct and external, according to Wohlin et al. (2000).

Conclusion validity concerns the relationship between the treatment and the outcome. Robust statistical techniques are used, measures and treatment implementation are considered reliable. The data were plotted and tested to check if it was normally distributed. In all cases, the data could not be concluded to be normally distributed and, thus, non-parametric tests were used. However, a threat is low statistical power, since only 16 subjects were used.

Furthermore, we have tried to increase the reliability of measures by conducting a pilot experiment and thereafter adjusting the wording and instrumentation. Another issue is that objective measures, e.g., time-consumption, are more reliable than subjective ones, e.g., ease of use and accuracy. However, the subjective measures are very important in this experiment and therefore we have chosen to include them. The experiment took place during one single occasion and therefore the implementation and setting are not a threat in this case.

Internal validity concerns the relationship between the treatments and the outcome of the experiment. The internal threats that may have affected the experiment are the fatigued effect, testing and group pressure. The subjects could become fatigued during the experiment, which may affect the concentration. In particular, the subjects who perform the tasks with 16 requirements may get tired or bored. This has been checked in the analysis, by calculating the consistency index for PWC. There is no significant difference in consistency for groups using different numbers of requirements (see Table 8). Hence, we draw the conclusion that the threat to the fatigue is low.

The testing threat is that the subjects get practice during the experiment and unconsciously get an opinion on the context using the first technique, which will affect the result for the second technique. At least when using PG first, it may affect the PWC performance. In Table 9, the order effect on consistency is analysed. There is no statistical difference in consistency depending on the order. Hence, this indicates that learning effects have not affected the experiment.

The third internal threat is the group pressure that may affect the subjects to rush through the task. In Section 3.2.4, there is an analysis of the correlation between the time used by the subjects and the consistency. The data indicates that the time-consumption has not affected the consistency of the prioritisation.

Construct validity concerns the relation between theory and observation. One threat in the design has been observed. It would have been valuable to start the session with an introduction explaining each requirement in the prioritisation to clarify their meaning. However, the subjects had their own interpretation of the requirements, which was the same throughout the experiment and therefore this should not affect the result.

External validity concerns whether the outcome of the experiment can be generalized to the population. Threats to external validity limit the generalisability of the experiment to industrial practice. The subjects are sampled from software engineering PhD students. Hence, the outcome of the experiment can be generalized to this group. In addition, for this experimental context it is likely that this group would perform equally to the requirements engineers and product managers who are intended to use the techniques in practice. The subjects are familiar with the application domain (mobile phone requirements) and several of the participants had prior working experience. The difference between industrial professionals and students in their final years has been considered small in other studies (Höst et al., 2000; Runeson, 2003). Furthermore, if a student experiment shows that one technique is better than another it is rather unlikely that professionals would come to the opposite conclusion (Tichy, 2000).

As most experimental conditions, the time is an important factor. In order to reduce the time needed for the experiment, the number of prioritised requirements is rather few. In most real cases, the total number of requirements is higher and therefore the results found in this study may be valid if the prioritisation is performed on a subset of the requirements. This may be the case e.g., if only the newly arrived requirements are prioritised or only the requirements for a certain sub-system. It is difficult to judge whether extending the number of requirements would lead to the same result. Therefore, future replications and case studies have to be made in order to draw conclusions when more requirements are used.

As the requirements used in this experiment are rather independent, they may have been easier to prioritise than is usually the case in industry. For example, the time required to perform the prioritisation would probably be larger in an industrial case due to more difficult trade-offs and dependencies between requirements. Requirements dependencies can require a group of requirements to be selected for a release instead of individual ones. This has not been investigated in the experiment.

A recent study investigated different criteria for selecting requirements for a certain release (Wohlin and Aurum, 2005). The results indicate that technical concerns, such as requirements dependencies, are less important than management-oriented criteria when deciding which requirements to select for a project or release. Therefore it is likely that requirements dependencies would have a relatively small effect on the results in an industrial case. We believe that these results may be used as a pilot for identifying trends before conducting a study in industry (Berander, 2004).

In summary, the main threats to the validity are that fewer, and more independent, requirements were used than in most industry cases. Hence, future replications are needed in order to reduce these threats. We believe that the other threats are under control. However, one mistake was made during the experiment. The scales "more than" and "less than" in the PWC sheets were accidentally switched so that it could be interpreted in the opposite way than was intended (see Figure 5). This caused some confusion during the experiment. However, the interpretation was explained and clarified and therefore this should not be considered as a threat to validity.

3.2.4 Data Analysis

The analysis of the experiment was divided between two independent researchers, in order to save time and to perform spot checks so that the validity could be further improved. The analysis was performed with Microsoft Excel^{TM}, the computing tool MATLAB^{TM} and the statistical analysis tool StatView^{TM}.Two different scales were tried for the PWC analysis: 1 ~ 5 and 1 ~ 9. According to Zhang and Nishimura (1996) the scale 1 ~ 5 is better than 1 ~ 9 at expressing human views and therefore the scale 1 ~ 5 was used when compiling the prioritisation ranking lists.

Furthermore, Saaty (1980) has calculated *random indices* (RI) that are used in the calculation of the consistency ratios. Unfortunately, this calculation only includes 15 factors while this experiment included as many as 16 factors, i.e., requirements. Therefore, the RI scale was extrapolated and the RI for 16 requirements was set to 1.61.

H1: Time-Consumption. The time to conclude the prioritisation is larger with PWC than with PG, for both criteria. As Table 2 shows, the difference in time between the two techniques is 6.1 minutes for 8 requirements, which corresponds to an increase of 43%, and 14.7 minutes for 16 requirements, which corresponds to an increase of 55%. Thus, for 16 requirements, it takes more than twice as much time to use the PWC compared to the PG, while for 8 requirements, the difference is a bit smaller.

The time increase in percent from 8 to 16 requirements for PWC is 88%, while the same for PG is only 48%. Thus, a larger number of objects to prioritise affect the time-consumption for PWC more than for PG, at least when using 8 and 16 requirements.

This can also be seen in Figure 7, where the median values are higher for PWC than for PG, and the difference between 8 and 16 requirements is larger for PWC than for PG. Additionally, the box plot indicates that the subjects' time to conclude the prioritisation with PWC are more dispersed.

Table 2. Average time-consumption for the prioritisation

Nbr of requirements	**Criteria**	**PG**	**PWC**	**Difference**
8	Value	3.6 min	7.8 min	4.2 min
	Price	4.5 min	6.4 min	1.9 min
Total		8.1 min	14.2 min	6.1 min
%				43%
16	Value	6.5 min	12.6 min	6.1 min
	Price	5.5 min	14.1 min	8.6 min
Total		12.0 min	26.7 min	14.7 min
%				55%
% increase		48%	88%	

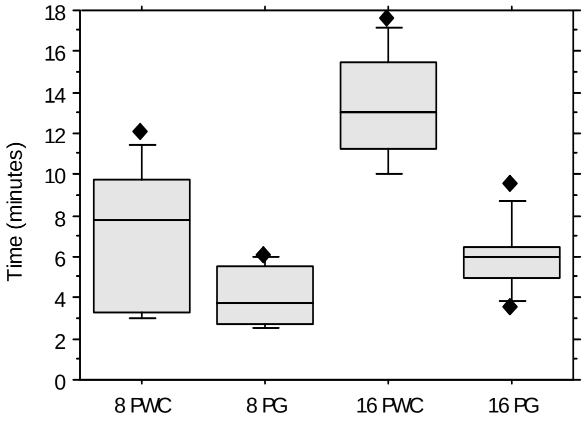


Figure 7. *Box plots of the time spent on prioritisation*

As Table 3 shows, the subjects have in average used less time per requirement when they had more requirements to prioritise. It is particularly interesting to see that it takes less time per requirement to perform PG partitioning with 16 requirements than with 8. One could expect that it should be more complex to perform PG with more requirements but this result show that more requirements tend to speed up the prioritisation per requirement. However, there might be a breakpoint when the number of requirements is too great and it becomes hard to get the valuable overview of the PG cards.

Four hypothesis tests were performed, for 8 and 16 requirements respectively, and one for each criterion. The frequency distribution was plotted in histograms to check the distribution. Due to the not normally distributed sample, we chose a non-parametric test, the *Wilcoxon test* (Siegel and Castellan, 1988). The hypothesis tests show that on the 5%-level there is a significant time difference for three of the four cases. This is illustrated in Table 4, where the p-value is lower than 5% in three of the four cases. Thus, the first null hypothesis is rejected for these cases.

Table 3. Average time-consumption per requirement

Nbr of requirements	PG	PWC
8	30.5 s/requirement	53.5 s/requirement
16	22.5 s/requirement	50.0 s/requirement

Table 4. Wilcoxon tests for the time difference

Nbr of requirements	Criteria	Wilcoxon p-values
8	Value	0.0251
	Price	0.1159
16	Value	0.0209
	Price	0.0117

H2: Ease of Use. Immediately after the experiment, the subjects filled out the first post-test that, among other things, captured the opinions of the techniques' ease of use. Among the 16 subjects, 12 found PG easier or much easier to use than PWC. Only 3 found them equally easy and 1 stated that PWC was easier to use, see Table 5. Hence, 75% of the subjects found PG easier to use.

This was tested in a *Chi-2 test* (Siegel and Castellan, 1988) by comparing the number of answers in favour of PWC to the number of answers in favour of PG. It turned out that there is a statistically significant difference, as p = 0.0023. Thus, the second null hypothesis is rejected.

It seems as if the subjects prioritising 16 requirements are a bit more sceptical to PG than those prioritising 8 requirements. This could indicate that the more requirements the more difficult to keep them all in mind.

H3: Accuracy. Directly after the experiment, the subjects performed the first post-test that captured which technique the subjects *expected* to be the most accurate. As Table 6 illustrates, a majority of the subjects expected PG to be better, while less than a fifth expected PWC to be better.

Table 5. Results from the first post-test: Ease of use

Nbr of requirements	PG Much easier	Easier	Equally easy	Easier	PWC Much easier
8	4	3	1	0	0
16	4	1	2	1	0
Total	8	4	3	1	0
Total %	50%	25%	19%	6%	0%

Table 6. Results from the first post-test: Expected accuracy

Nbr of requirements	Favour PG	Equal	Favour PWC
8	4	3	1
16	5	1	2
Total	9	4	3
Total %	56%	25%	19%

In order to evaluate which technique that gave the most accurate results, a second post-test was filled out by the subjects. This was done a few weeks after the experiment was performed, when the analysis was finished.

The most common opinion among the subjects was that PG reflects their views more accurately than PWC. This is shown in Table 7 where 47% of the subjects were in favour of PG and only 28% were in favour of the PWC. This is, however, not statistically significant, p = 0.2200 with a Chi-2 test, so it cannot be determined if there is a difference between the techniques' accuracy. Thus, the null hypothesis is not rejected. Half of the ones that have stated that both techniques are equally accurate actually had the same order in the lists.

An interesting observation is that this implies that PG was actually not as good as the subjects expected even if most subjects preferred PG to PWC.

Consistency Ratio. The consistency ratio (CR) describes the amount of judgement errors that is imposed during the pair-wise comparisons. The CR is described with a value between 0 and 1 and the lower CR value, the higher consistency. Saaty (1980) has recommended that CR should be

Table 7. Results from the second post-test: Perceived accuracy

Nbr of requirements	Criteria	Favour PG	Equal	Favour PWC
8	Value	6	2	0
	Price	1	3	4
16	Value	4	1	3
	Price	4	2	2
Total		15	8	9
Total %		47%	25%	28%

Table 8. Mean consistency ratios

Nbr of requirements	Criteria	Scale 1 ~ 5
8	Value	0.106
	Price	0.101
16	Value	0.082
	Price	0.120

lower than 0.10 in order for the prioritisation to be considered trustworthy. However, CR exceeding the limit 0.10 occurs frequently in practice (Karlsson and Ryan, 1997).

The CR limit above is only valid for the scale 1 ~ 9, and in this experiment the scale 1 ~ 5 was used instead. Therefore, the limit for acceptable CR will be lower. The average consistency ratios for scale 1 ~ 5 are presented in Table 8.

The frequency distribution for the consistency was plotted in histograms to check the distribution. The data were not normally distributed and therefore we chose a non-parametric test. The Wilcoxon test resulted in p > 0.30 for both criteria. Therefore, it cannot be proved, on the 5%-level, to be a significant difference in consistency depending on the number of requirements prioritised.

In order to investigate if the time spent on each comparison affects the consistency, the correlation between these parameters was calculated. The *Spearman rank-order correlation* coefficients indicate no correlation between the time and the consistency, as the correlation varies between -0.40 and 0.20. According to Siegel and Castellan (1988), the absolute value of the correlation coefficient should be greater than 0.738 in order for the correlation to be considered significant in this case. Hence, the consistency is not particularly influenced by the time spent on prioritisation.

Order Effects. There is a chance that the order in which the two techniques are used can influence the result. Table 9 shows that the mean consistency ratio is a bit lower for the subjects who used PG before PWC. This may indicate that using PG can provide an image of ones preferences that are not possible to get from using PWC. Therefore it may be easier to be consistent when PG precedes PWC.

However, the hypothesis tests show that the difference is not significant on the 5%-level. Due to the not normally distributed sample,

Table 9. Order effect on consistency

Mean consistency	**PWC-PG**	**PG-PWC**
Value	0.107	0.082
Price	0.119	0.102

we chose a non-parametric test, the *Mann-Whitney test* (Siegel and Castellan, 1988). The p-values are larger than 0.6, and therefore we cannot confirm a significant difference depending on the order.

A set of significance tests was also conducted investigating the order effect on time-consumption. Neither of the cases (with 8 or 16 requirements or with Value or Price criterion) showed a significant difference in time depending on the order in which the techniques were used. This finding validates that the experiment analysis has not suffered from any order effects, neither regarding time nor consistency.

Qualitative Answers. In the post-test performed right after the experiment, the subjects had the opportunity to answer some optional questions about their general opinion. Opinions about PWC include "effort demanding but nice", "it feels like a blackbox wherein you pour requirements", "good but boring", "it feels like you lose control over the prioritisation process", and "straightforward". Opinions about PG are for example "fast and easy", "lets the respondent be creative", "intuitive", "prone to errors", "good overview", and "logical and simple". These opinions correspond well to the results of the captured subjective dependent variables: ease of use and expected accuracy, discussed in prior sections.

3.2.5 Results

The main results are that the PG technique is superior to the PWC regarding the two variables time-consumption and ease of use, while it could not be determined which technique that has the highest accuracy.

Two groups prioritised 8 and 16 requirements, respectively, in order to investigate if there is a breakpoint between 8 and 16 where one of the methods is more efficient than the other. It was suspected that a greater number of requirements would eliminate the valuable overview in PG, since it would be difficult to keep all requirements in mind. However, this experiment only shows an insignificant tendency of less overview affecting the ease of use when prioritising 16 requirements (see Table 5). Therefore, it is suspected that the breakpoint is at an even higher number of requirements.

Another interesting observation in this experiment was that the time-consumption did not affect the consistency in PWC (see Section 3.2.4). One could assume that if someone rushes through the comparisons, the consistency would be poor. However, these are only initial results and with another set of objects to prioritise, the results might be different.

The objective measure *total time-consumption* is higher for PWC than for PG both in our study and in the one by Karlsson et al. (1998). On the other hand, PWC was given a higher rank than PG regarding the subjective measures ease of use and fault tolerance in the study by Karlsson et al. (1998). Our experiment shows that PG is easier to use than PWC. This difference in result may be due to differences in methodology. While our study is a controlled experiment with 16 participants, Karlsson et al. (1998) is based on an evaluation by three individuals who discussed their opinions. The result regarding time-consumption is considered reliable, while the difference regarding ease of use indicates that additional studies need to be performed in order to further understand the strengths and weaknesses of these techniques.

Karlsson et al. (1998) suggested a combination of the two techniques Priority groups and PWC, in order to use the PWC with a reasonable amount of effort. Using PWC on the three priority groups, separately, would decrease the number of comparisons. Another possibility is to use PWC only on those requirements that end up in the middle priority pile. This would imply that PG, or Priority groups, is used first, to divide the requirements into three groups. The high priority group of requirements will most certainly be implemented, the low priority group will be postponed and looked into in a following release, while the ones in the middle need special treatment to determine the outcome.

This approach agrees with what Davis (2003) has written about the *requirements triage* where he recommends requirements engineers to focus on the difficult requirements and skip the ones that will either be implemented or rejected anyway. In this manner, PWC can be used on the requirements that are difficult to estimate and need a more precise scale for determining its cost and value. The technique's ratio scale and fault tolerance would then come to its right.

3.3 Experiment 2

This section describes the second of the two experiments, the experiment planning, operation and analysis. Finally, the section is concluded by a discussion of the results. Much of the design in the first experiment have been reused in the second one, therefore several references are made to Section 3.2.

The motivation for the second experiment is that although the first experiment indicates that PG is superior to PWC, we suspect that PWC with tool-support may have certain benefits for practitioners. With tool-support it is possible to reduce the number of comparisons and to visualise the priorities. It may also be easier to use, as it guides the decision-maker during the prioritisation process. We believe that the PWC would benefit more than PG from tool-support, and therefore we chose to investigate the tool-supported PWC (TPWC) and compare it with PG.

3.3.1 Hypotheses and Variables

The goal of the second experiment is to compare two prioritisation techniques and to investigate the following null hypotheses:

$H_0 1$: The average time to conclude the prioritisations is equal for both techniques, PG and TPWC.

$H_0 2$: The ease of use is equal for both techniques, PG and TPWC.

$H_0 3$: The accuracy is equal for both techniques, PG and TPWC.

The alternative hypotheses are formulated below:

$H_A 1$*:* The average time to conclude the prioritisations is not equal for both techniques, PG and TPWC.

$H_A 2$*:* The ease of use is not equal for both techniques, PG and TPWC.

$H_A 3$*:* The accuracy is not equal for both techniques, PG and TPWC.

The independent variables are the techniques PG and TPWC and the dependent variables are the same as in the first experiment, i.e., *average time to conclude the prioritisations*, *ease of use* and *accuracy*.

The time-consumption was captured by each subject by noting their start and stop time for each task, the ease of use was measured by a questionnaire which was filled out by all subjects after the experiment, and the accuracy was measured by conducting a post-test a few weeks after the experiment similarly to Experiment 1.

3.3.2 Experiment Design

The second experiment was also carried out with a *repeated measures design, using counter-balancing*, i.e., all subjects used both techniques. The subjects were 30 MSc students (25 male and 5 female) in their final year, taking an optional requirements engineering course. The experiment was conducted within a compulsory laboratory session in the area of requirements prioritisation. The session was conducted for teaching purposes and gave the students an opportunity to try out and compare two commonly known prioritisation techniques. No pre-test was performed, so the participants were randomly assigned to perform the tasks in a certain order.

The experiment was divided into two separate occasions with 20 subjects at the first session and 10 at the second. Both sessions were guided by two teachers. Before the experiment the participants were given an introduction to the tool by conducting a comprehensive tutorial.

PG was used in the same manner as in the first experiment. The TPWC used a RM tool with pair-wise comparisons as prioritisation technique. The participants conducted the number of comparisons required by the stopping rules in the tool (approximate size *2n*), and could revise comparisons when inconsistency was indicated by the tool. Note that the tool was used only as an approach to prioritisation, i.e., the visualisation possibilities in the tool were not investigated.

Two post-tests, which are described below, were performed similarly to the first experiment in order to capture the dependent variables. Figure 8 illustrates the activities performed in the second experiment.

Execution. The experiment took place in a computer laboratory room during a compulsory laboratory session. The manual technique PG was used in the same room but the students could move to empty desks.

For each subject an experiment kit had been prepared, consisting of the PG cards and a personal instruction regarding the order to perform the tasks. Each subject also had a personal login to the prioritisation tool.

Data were mainly collected through post-tests. The PG priority piles were attached with a paper clip and handed in, while the TPWC lists were compiled by the researcher after the session by extracting the information needed from the RM tool. Each subject noted the start and stop time in the post-test conducted right after the experiment, as well as their opinion on ease of use. Then, the second post-test captured the accuracy through a

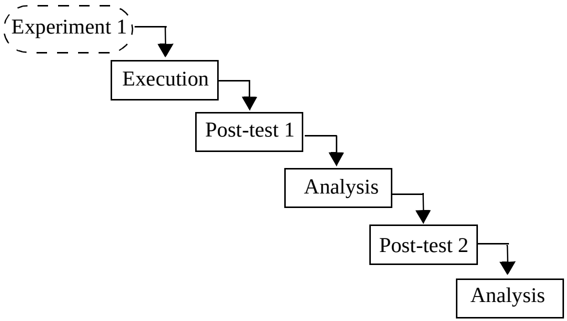


Figure 8. *Activities conducted in Experiment 2*

blind-test a few weeks later. The subjects were given 2 hours to perform the tasks, including the introductory tutorial for the tool.

The design of the second experiment is very similar to the first one, since it was intended to investigate the same hypotheses. Thus, the main difference was that the PWC was tool-supported in the second experiment.

Furthermore, since the first experiment showed that the number of requirements did not affect the outcome of the first experiment, it was decided to have all participants prioritise between 16 requirements. The same mobile phone requirements were used, as well as the same criteria Value and Price. The counter-balancing design is illustrated in the appendix, Table B.

Post-Test 1. The subjects were asked to fill out the same post-test as in experiment 1 after they had handed in their experiment kit. This was made in order to capture the subjects' opinions right after the experiment.

Post-Test 2. A few weeks after the experiment, the subjects were, in a second post-test, asked to state which technique they found most accurate. This was conducted as a blind-test in the same way as for the first experiment, but the lists corresponded to the results from the two techniques PG and TPWC.

3.3.3 Threats to Validity

This section discusses the threats to validity for the second experiment. The same four classes of validity threats as for the first experiment are considered in this section.

Conclusion validity. As in the first experiment, the statistical techniques, measures and treatment implementation are considered reliable. Both objective and subjective measures are used. The student group is a homogeneous group, with similar background and education.

Internal validity. The internal threats in the second experiment is the fatigue effect, mortality and the instrumentation. The fatigue threat is present since one of the sessions took place after office hours. However, there was no disturbance during the performance.

Furthermore, the subjects could be influenced by the first priority list, and unconsciously prioritise in a similar manner when producing the second priority list. On the other hand, when conducting the pair-wise comparisons, it is difficult to use knowledge from another prioritisation, which reduce the threat. In addition, in the first experiment, the threat was estimated as low, and there are no indications that it would be higher in the second experiment.

Another threat is the mortality effect. This is small, but present since one of the subjects was absent during the second post-test and therefore one data point is missing.

There is also a potential instrumentation threat. The TPWC technique was not used to visualise the priority list since we intended to conduct the second post-test with a comparison of the lists from the two techniques. However, the PG technique directly results in a priority list and it can therefore not be hidden. Therefore, some subjects may have remembered the priority order from the PG and could thereby identify which of the lists in the second post-test that correspond to which technique. This may also have been the case in the first experiment, but then the time between the experiment and the second post-test was longer, which reduces the risk of remembering. However, we believe that the subjects chose a list based on perceived accuracy and not based on remembering which list the priorities come from.

Construct validity. The experiment would need another set of requirements to perform the prioritisation on in order to be able to discover if the results are the same, or if the set of requirements have affected the results. Testing and treatment may interact. When the subjects know that the time is measured, it is possible that they get more aware of the time they spend, and thus the time-consumption is affected. However, they were not aware of the other two measures when conducting the experiment, so only the time can have been affected.

External validity. The subjects in the experiment is a homogenous group. This improves the conclusion validity, but makes it more difficult to generalise the result to a broader population. This issue was discussed in Section 3.2.3 for the subjects in the first experiment and the same discussion is valid for the subjects in this experiment.

As discussed in Section 3.2.3, the small number of requirements decreases the possibility to generalise to cases where a higher number of requirements is prioritised.

In summary, the main threat in this experiment is the instrumentation threat and that it is difficult to generalise to situations where a larger set of requirements are prioritised.

3.3.4 Data Analysis

This section presents the analysis and results from the second experiment. The analysis was performed by two researchers using Microsoft Excel^{TM}, the computing tool MATLAB^{TM} and the statistical analysis tool StatView^{TM}.

H1: Time-Consumption. The first hypothesis regards the time needed to perform the prioritisation. As can be seen in Table 10 the average time required is lower for both criteria when using the TPWC. In fact, TPWC required 17% less time than PG.

As can be seen in the box plots in Figure 9, where the times for both criteria are added, the median values are higher for PG than for TPWC. The times for PG are also more dispersed.

Normal probability plots indicated that the data were not normally distributed. Therefore, it was decided to use non-parametric tests during analysis. The difference in time is significant on the 5%-level as the Wilcoxon test results in p-values below 0.04. Therefore we can draw the

Table 10. Average time-consumption (in minutes)

Criteria	**PG**	**TPWC**	**Difference**
Value	5.8	4.8	1.0
Price	5.5	4.6	0.9
Total	11.3	9.4	1.9
%			17%

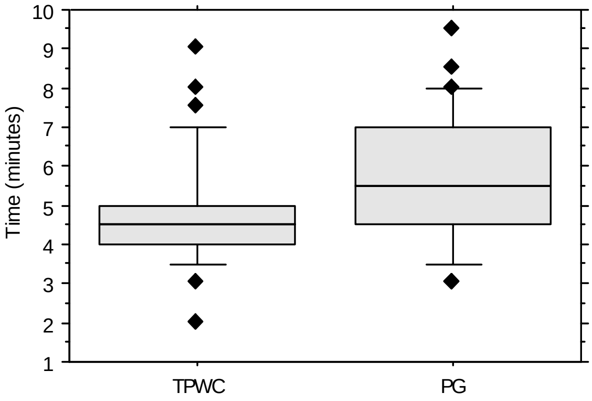


Figure 9. *Box plots for the time spent on prioritisation*

conclusion that the TPWC technique is a faster technique than the PG, i.e., the null hypothesis is rejected.

H2: Ease of Use. After using both techniques, the participants handed in a post-test answering the question "Which technique did you find easiest to use?". In total, 10 of the 30 subjects found the PG easier or much easier to use, while 16 pointed out TPWC as easier or much easier. As can be seen in Table 11, this corresponds to that 33% found PG easier, while 53% found TPWC easier. 4 of the subjects, i.e., 13%, found the techniques equally easy to use.

A Chi-2 test shows that the difference between the number of subjects that found PG easier and the number of subjects that found TPWC easier is not significant, p = 0.2393. Thus, the null hypothesis cannot be rejected.

H3: Accuracy. The first post-test also captured which technique the participants expected to be the most accurate. As can be seen in Table 12, 77% of the subjects expected the TPWC to be more accurate than the

Table 11. Results from the first post-test: ease of use

Ease of use	PG Much easier	Easier	Equally easy	Easier	TPWC Much easier
	1	9	4	11	5
%	3%	30%	13%	37%	16%

Table 12. Results from the first post-test: Expected accuracy

Expected accuracy	PG Much more accurate	More accurate	Equally accurate	More accurate	TPWC Much more accurate
	0	5	2	14	9
%	0%	17%	7%	47%	30%

PG. Thus, a majority of the subjects found the tool trustworthy after using it.

However, the second post-test investigated which of the techniques the subjects found most accurate by conducting a blind-test where the subjects were given their priority lists from both techniques. Due to absence, only 29 of the 30 participants filled out the second post-test. As can be seen in Table 13, where both criteria are added, 50% found the PG lists more accurate, while 37% found the TPWC lists more accurate. 12% found the priority lists equally accurate. This difference is not statistically significant, as the p-value turned out to be 0.3270 in a Chi-2 test.

Thus, the TPWC did not get as high accuracy as expected, while PG turned out to be more accurate than expected. The null hypotheses cannot be rejected.

Order Effects. A set of significance tests was used to investigate whether or not there was a significant order effect on the time-consumption. The time-difference depending on the order in which the techniques were used, was investigated with a Mann-Whitney test. The test did not indicate a significant time-difference ($p > 0.10$ for Value and $p > 0.90$ for Price) and therefore we cannot show a significant order effect. Another set of tests investigated the effect on time-consumption depending on the order in which the criteria were used. In this case, a Wilcoxon test was

Table 13. Results from the second post-test: Perceived accuracy

Perceived accuracy	PG Much more accurate	More accurate	Equally accurate	More accurate	TPWC Much more accurate
Value	0	15	1	12	1
Price	1	13	6	8	1
Total	1	28	7	20	2
%	2%	48%	12%	34%	3%

used and could not show a significant order effect ($p > 0.60$ for both TPWC and PWC).

A third significance test was used to investigate if the occasion (afternoon or evening) affected the time-consumption. The Mann-Whitney test indicates no significant difference in time-consumption ($p > 0.40$ for Value and $p > 0.10$ for Price). Thus, we cannot determine any significant effect on the time-consumption depending on different orders.

Qualitative Answers. Some personal opinions on the experiment and the two techniques were also collected. Among the positive views on TPWC are "TPWC is probably better than PG for larger projects" and "TPWC is easy to use". There were also some opinions in favour of the PG, "TPWC makes it difficult to keep focus with many requirements" and "PG gives a better overview of the requirements".

3.3.5 Results

One of the main reasons for conducting the second experiment was that it was suspected that the manual PWC in the first experiment would benefit from tool support so that the drawbacks of e.g., high time-consumption could be reduced. The main result from the second experiment is that the Tool-supported PWC is a faster technique than the PG. Thus, the first null hypothesis could be rejected. However, although there are more subjects finding TPWC easier to use than PG, the difference is not statistically significant. A difference in accuracy could not be determined either. Thus, the second and third null hypotheses could not be rejected.

There were no significant order effects depending on e.g., the order in which the techniques were used. However, using PG first and then TPWC on the separate piles would still decrease the necessary time-consumption, although the time reduction would not be as large as in the case with PG and manual PWC. This is due to the fact that TPWC only require approximately $2n$ comparisons.

3.4 Discussion

Prioritisation is a very important activity in requirements engineering because it lays the foundation for release planning. However, it is also a difficult task since it requires domain knowledge and estimation skills in order to be successful. The inability to estimate implementation effort and predict customer value may be one of the reasons why organisations use ad hoc methods when prioritising requirements. For a prioritisation technique to be used it has to be fast and easy to manage since projects often have limited time and budget resources.

The experiments presented in this study have investigated the time-consumption, ease of use and accuracy for different prioritisation techniques. But when deciding which prioritisation technique to use in an organisation there are several other aspects to take into consideration. The technique has to be supported by the software process and other project activities. For example, the PG may be successful if the overall development approach is agile and requirements are already written on e.g., story cards (Beck, 2005). On the other hand, if a RM tool is used and requirements are already stored in the tool, it is evidently natural to use it as a means for prioritisation as well. Thus, it is necessary to consider methods and tools for requirements prioritisation to be aligned with other methods and tools used in the organisation.

Another related issue is the necessary analysis effort that must be used in order to get a priority list from the conducted prioritisation. In PG, the result is in the form of ranked piles of cards, which need to be transformed into e.g., a Cost-value diagram in order to sufficiently visualise the trade-off between cost and value. The manual PWC requires plenty of analysis and matrix calculations before a priority list can be extracted and visualised. This analysis is not realistic to perform manually when the number of requirements grows. The PWC can provide additional information compared to PG, such as the consistency, and the data is on a ratio scale. The Tool-supported PWC has several different visualisation possibilities and the tool takes care of all calculations and displays the prioritisation in charts and diagrams. Thus, the required analysis differs between the techniques and it needs to be taken into consideration before deciding on a prioritisation technique.

Furthermore, minor investments are needed for the manual techniques since all resources required are pen and paper, while on the other hand more staff resources are needed to analyse the outcome, as discussed above. Commercial tools may be expensive and can be risky to rely on since anything from computer crashes to vendor bankruptcy can occur. On the other hand, it visualises priorities without any extra effort.

The generalisability of the study is limited due to the rather small sample and the specific context. Although the subjects may have opinions similar to decision-makers in industry, the context of mobile phone requirements may be a bit too simplistic. The main weakness is that mobile phone requirements are on a high level and rather independent, while requirements in a real case often have interdependencies. Industrial projects also have time and budget pressure to consider, which complicates the decision-making. It is possible that industrial experience would affect the results, although we believe that in a relative comparison between the techniques, it is likely that the results would be similar.

The validity of controlled experiments using students as subjects is often debated. In Carver et al. (2003) it is acknowledged that the higher level of control in controlled experiments compared to e.g., case studies may help researchers ascertain whether a phenomenon of interest has statistically significant effect. This may not be possible in an industrial case study, as the environment itself decreases the possibility to generalise to other companies. Furthermore, hardly any industrial software developer can afford to use two different technologies to evaluate which one is more effective. Instead, this kind of study can be carried out in an empirical study with students (Carver et al., 2003).

The first part of PG is based on numeral assignment as each requirement is assigned to one of the three piles. This approach is similar to the manner used in many organisations, i.e., classifying each requirement as having high, medium or low priority. In an industrial situation it is common that most requirements are classified as high (Karlsson, 1996). To avoid that, some constraints might be needed, such as imposing each classification to include at least 25% of the requirements. It is, however, rarely sufficient to use only numeral assignment since the difference in importance of requirements assigned the same priority can be larger than the difference in importance of requirements assigned different priorities (Karlsson, 1996).

In practice, it is common that a larger number of requirements need to be prioritised. The results presented in this study may be valid when a sub-set of the requirements is prioritised. When the number of requirements grow, it is hard to get an overview. Therefore, visualisation becomes very important in order to share information. In a real project, it may also be more valuable to use the ratio scale in order to, in more detail, differentiate requirements from each other. Thus, it may not be sufficient to determine which requirement that is of higher priority, without knowing to what extent. This would speak in favour of the PWC techniques.

It is interesting to explore a possible extension to PG, providing it with a ratio scale. When the requirements have been ordered in a priority list using PG it would be possible to compare each requirement to the one below it in the list and assign a number to their internal relation. For example, one requirement can be estimated as being twice as important as the one below it in the priority list, and thereby their relation is set to two, and so on. In this manner, it would be possible to, with a reasonable amount of effort, provide PG with a ratio scale. More research needs to be conducted in order to determine the validity of this extension.

3.5 Conclusions

The main conclusion that can be drawn from both experiments is that the TPWC is superior to both PG and PWC regarding time-consumption. This may be due to the reduced number of comparisons in the tool compared to in the manual techniques. It can also be an effect of the increased support for the user as only one pair is displayed and it is therefore easier to stay focused. It would also be interesting to investigate how tool-support for PG would affect the results.

PG was regarded as easier to use than the manual PWC in the first experiment, while it could not be determined if either of the techniques TPWC or PG is easier to use, although a majority found the TPWC easier. This may be due to most subjects enjoying to use a tool-based technique more than a manual one.

A difference in accuracy could not be confirmed in either of the experiments although PG was preferred by most of the subjects in both experiments.

Although the generalisation of the presented experiments to industrial practice is not straightforward, the results are an important basis for the planning of industrial case studies. When companies want to find a prioritisation technique that suits their needs they can take the presented results into account when planning situated trials.

The presented experiment design could also be used on more subjects to get a larger data set and thereby a stronger basis for conclusions. There are, as discussed, several other prioritisation techniques that would be interesting to look into and compare to the presented techniques as well.

4. Evaluating the Practical Use of Different Measurement Scales in Requirements Prioritisation

This section presents the evaluation of measurement scales for requirements prioritisation, described in Paper 3. Early results from this study are presented in Paper 17.

4.1 Background

This section presents an empirical investigation, which was performed by analysing ratio scale prioritisation results gained from four different experiment assignments. Most of the 36 subjects were students or PhD students taking a requirements engineering (RE) course or a research methodology course. The subjects used Pairwise comparisons to prioritise requirements, so the resulting priorities are presented on a ratio scale. In two of the cases, the participants were supported by the RM tool Focal Point (Telelogic, 2006) during prioritisation. The results were analysed after the experiment assignments.

We would like to determine which situations that require the ratio scale and in which situations the ordinal scale is sufficient. Therefore we need a measure that can describe the characteristics of the ratio scale distribution so that results from different prioritisation sessions can be compared. The measure is called skewness and is the standard deviation for the difference between a ratio scale distribution and a baseline distribution. In addition we want to investigate how the cost-value approach can be used if prioritisation is performed on an ordinal scale. Therefore we attempt to answer the research questions below.

RQ1. How can we measure the skewness of a ratio scale prioritisation distribution?

RQ2. How can the cost-value approach be applied when the priorities are based on ranks?

This section is outlined as follows. Section 4.2 explains the empirical investigation and the data analysis. Section 4.3 discusses the results and the validity of the results. Section 4.4 concludes the study and provides some ideas for further work.

4.2 Methodology

This section presents the method used in our empirical investigation, and the analysis of the data. In order to investigate the skewness of the ratio scale distribution (RQ1) we need authentic ratio scale data results from real prioritisation sessions. These data can be compared to a baseline in order to get a measure of the skewness for each subject. Similarly, for the cost-value approach (RQ2), we need to compare ratio scale cost-value diagrams with ordinal scale cost-value diagrams in order to see if the requirements selection differs depending on the scale. It was decided to use the ratio scale data and reduce it to ordinal information to draw the ordinal cost-value diagrams for each subject. Thereby, we can compare ratio and ordinal scale diagrams based on the same data set. These data were obtained from experiments and assignments conducted at the university.

4.2.1 Data Collection

The research methodology is based on *archive analysis* as the data were produced for purposes other than this research (Robson, 2002). Four different data sets, containing data from 36 subjects, were investigated, see Table 14. Two of the data sets were obtained in experiment assignments conducted within a Masters' course. The other two sets were obtained from the experiment presented in Section 3.2.

Data set 1: First experiment assignment. The first data set was extracted from an experiment assignment in an optional RE course for Master's students. The purpose of the experiment assignment was to teach

Table 14. Outline of the four data sets

Data set	Setting	Number of subjects	Number of requirements	Cooperation/ Individual
1	Experiment assignment	8	21	Cooperation
2	Experiment assignment	12	21	Cooperation
3	RE experiment	8	8	Individual
4	RE experiment	8	16	Individual

the 25 students about the challenges of prioritising requirements and to allow the students to investigate a commercial RM tool. During the assignment, 2-3 students cooperated on one computer, as it is a realistic situation for decision-makers. In total, ten groups of students participated; thus ten priority results were obtained, hereafter called subjects. The subjects were asked to prioritise between 21 mobile phone requirements, such as Chat function, Wireless Application Protocol (WAP), Predictive text input (T9), etc. (see appendix, Table D). The tool utilises Pair-wise comparisons as prioritisation technique and has an algorithm that reduces the number of necessary comparisons, without jeopardising the consistency. The resulting priorities are presented as percentages on a ratio scale. The subjects were encouraged to choose two different criteria, one to maximise and one to minimise. Most subjects selected one criterion related to value, and one criterion related to cost. In the analysis after the experiment assignment, two of the ten subjects were removed, as their criteria were not consistent with either value or cost. The ratio scale data from the resulting eight subjects were then examined as described in Section 4.2.2.

Data set 2: Second experiment assignment. The second data set was obtained in a similar manner as the first one. The same RE course was given a second semester to 26 students. Two of the students worked alone and the rest worked in pairs, resulting in 14 priority results, hereafter called subjects. The same tool was used, as well as the same requirements in the prioritisation task. The subjects were encouraged to select the criteria cost and value but as two of the subjects choose other criteria they were removed from the analysis. As we anticipated the results to be used for research purposes, we also posed some qualitative questions after the session. These qualitative results are presented in Section 4.2.2.

Data set 3 and 4: RE experiment. The third and fourth data sets are based on an experiment conducted to compare two different requirements prioritisation techniques: Planning game and Pair-wise comparisons. The experiment tested the difference between the techniques regarding time-consumption, ease of use, and accuracy, see Section 3.2. In this case, the Pair-wise comparisons were performed manually, i.e. no tool was allowed. The 16 subjects worked individually, as the task was performed for experimental purposes. The experiment yielded two different sets of data as half of the subjects prioritised between eight requirements and half of the subjects prioritised between 16 requirements, see Table C in the appendix.

4.2.2 Data Analysis

The priority results from the experiment assignments and the prioritisation experiment were investigated in two different ways. First, the results were analysed to investigate the skewness of the ratio scale priorities. Secondly, the ratio scale data were reduced to ranks, i.e. ordinal scale data, and investigated in cost-value diagrams. Finally, some qualitative results from the second experiment assignment are presented.

Evaluation of skewness. The first research question concerns possibilities of measuring the skewness of the ratio scale distribution. In this section we define a skewness measure based on the standard deviation for the difference between a ratio scale distribution and a baseline distribution. A more skewed distribution indicates that the person performing the prioritisation has given much larger weights to some of the requirements than others. A less skewed distribution indicates that the differences in priorities between requirements are not very large. In that case it could have been sufficient to use the ordinal scale since the distribution could be approximated with a linear distribution. Therefore, we use the linear distribution as a baseline. The measure of skewness can be used to determine in which situations the ratio scale is needed and when the ordinal scale is sufficient.

A bar chart illustrates the result of a prioritisation through bars of different length. The length of each bar represents the ratio for each requirement, thus the total value of all requirements adds up to 100%. Unlike the cost-value diagram, the bar chart shows the result for one criterion at a time.

The left bar chart in Figure 10 illustrates the bar chart from a ratio scale prioritisation for one of the subjects. The distribution is clearly skewed and the top requirement accounts for approximately 30% of the total value. If we want to evaluate how skewed this distribution is, we can compare it to a baseline bar chart where the difference between adjacent bars is equal all over the chart. This is shown in the right bar chart in Figure 10. Note that the requirements have the same ranks in both charts, but the right one is transformed to the baseline distribution. Imagine if the right one were the result from using a ratio scale technique. Then it

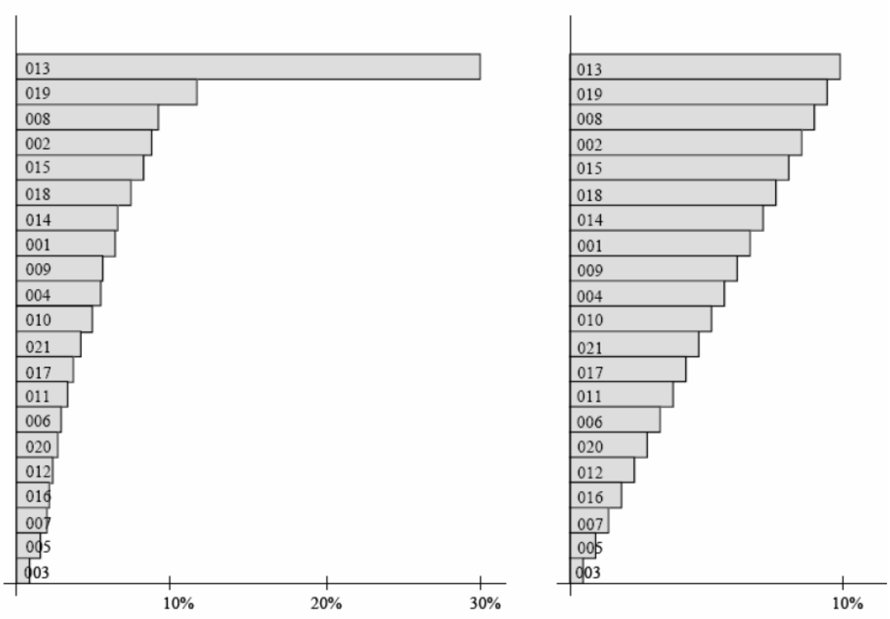


Figure 10. *Comparison between skewed and linear bar charts*

would have been more efficient to use an ordinal scale technique with ranking, as the difference in importance between requirements is the same all over the chart. Therefore we use the right bar chart in Figure 10 as the baseline when evaluating the skewness of the ratio scale bar chart.

PROCEDURE. To evaluate the skewness of a ratio scale prioritisation distribution we want a measure of the characteristics of the ratio scale distribution. This measure is obtained by comparing the ratio scale bar chart to the linear one, as shown in Figure 10, in the following manner. Suppose the value of the lowest ranked requirement in the linear bar chart account for k% of the total value. The second lowest ranked requirement would then account for 2*k% of the value. In a similar manner, the highest ranked requirement account for N*k% of the value, if there are N requirements in total. Thus, multiplying the constant k with the rank yields the linear equivalent to the priority. The difference between adjacent requirements is the same all over the chart, and equal to the constant k%. In order for the total value of all requirements to add up to 100%, the statement below can be used.

$$k + 2k + 3k + \ldots + Nk = 1$$

Using this equation, it is possible to calculate the constant k:

Eq 1. $$k = \frac{1}{1 + 2 + 3 + \ldots + N}$$

From these assumptions, we can calculate the skewness of the ratio scale data by calculating how much the ratio bar deviates from the constant k multiplied by its rank. This difference was calculated for each requirement and then the standard deviation, i.e. the skewness, was calculated as follows:

Eq 2. $$\text{Skewness} = \sqrt{\frac{\sum(\text{DiffForEachReq})^2}{N-1}}$$

Since the values of the ratio scale prioritisation often are presented in percentages, we have chosen to use percentages in the calculations.

EXAMPLE. For the example in Figure 10, the highest ranked requirement accounts for 28.7% of the value according to the ratio scale. To calculate the value of the highest ranked requirement in the linear distribution, we need to calculate the constant k with Equation 1, where N=21.

$$k = \frac{1}{1 + 2 + 3 + \ldots + 21} = 0,00433$$

Thus, the highest ranked requirement accounts for N*k%, i.e. 21*0.00433=9.1%. Consequently, the difference between the ratio scale distribution and linear distribution is 28.7%–9.1%=19.6% for this requirement. This difference is summed up for all the requirements and used in Equation 2 to calculate the skewness. In this example the skewness is 4.6%.

RESULT. The skewness was calculated for all subjects with Equation 2 and is presented in the tables below.

Table 15. Skewness for data set 1

	A1	A2	A3	A4	A5	A6	A7	A8	*Av*
Value (%)	4.6	1.3	1.5	1.0	2.0	4.3	1.3	2.7	*2.3*
Cost (%)	4.0	3.9	2.5	0.5	2.3	4.8	1.1	1.2	*2.5*

As seen in Table 15, Subject A4 has the smallest skewness for both criteria and thus the least skewed distribution.

Table 16. Skewness for data set 2

	B1	**B2**	**B3**	**B4**	**B5**	**B6**	
Value (%)	1.1	1.9	1.9	1.9	1.5	1.0	
Cost (%)	1.0	4.3	1.2	1.7	0.9	2.4	
	B7	**B8**	**B9**	**B10**	**B11**	**B12**	***Av***
Value (%)	1.3	1.1	1.2	2.9	1.0	2.8	*2.0*
Cost (%)	1.6	2.1	1.2	1.1	1.0	1.5	*1.6*

As Table 16 indicates, subject B11 has a small skewness for both criteria, while, e.g., subject B2 has a rather large skewness. The average skewness is smaller for the subjects in the second experiment assignment than in the first for both criteria.

Table 17. Skewness for data set 3

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	***Av***
Value (%)	3.5	2.3	3.6	2.7	6.2	3.2	3.1	2.2	*3.4*
Cost (%)	2.0	5.5	2.3	2.4	5.8	1.3	3.7	2.0	*3.1*

Among the subjects in Table 17, C5 has the most skewed distribution.

Table 18. Skewness for data set 4

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	***Av***
Value (%)	1.7	0.9	1.5	0.6	1.9	1.0	3.0	2.1	*1.6*
Cost (%)	1.2	3.5	1.5	0.3	0.9	1.5	1.0	1.9	*1.5*

Subject D4 has the lowest skewness among the subjects in Table 18.

ANALYSIS. The average skewness is rather equal for both criteria within each of the four data sets. Thus, the skewness seems to be independent of the criteria, at least for these criteria and these data sets. The size of the average skewness varies between the data sets. This can be explained by the different numbers of requirements in the different settings, and the difference in application, i.e. manual as opposed to tool-supported techniques.

The skewness seems to vary among the subjects. However, most subjects have rather similar skewness values for both used criteria. In fact, the correlation between the skewness for value and the skewness for cost is 0.37, p=0.028, see also Figure 11. This could imply that some people

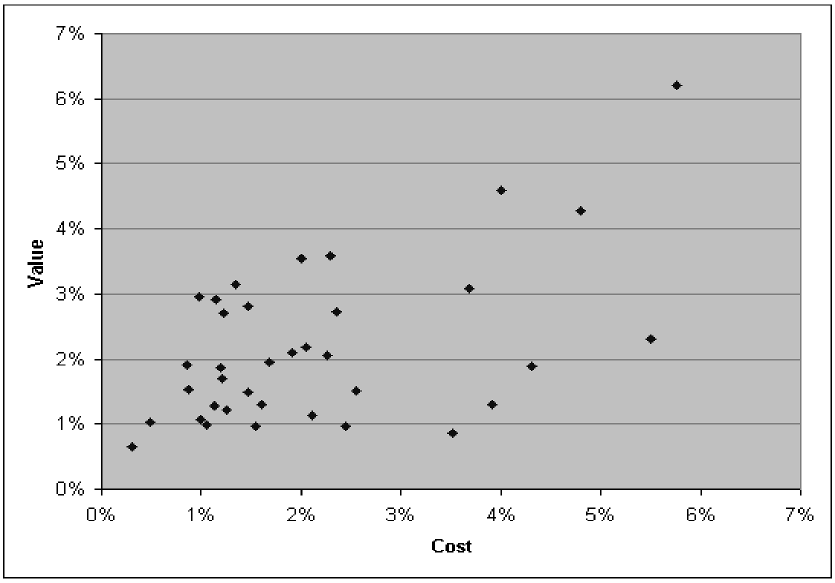


Figure 11. *Correlation between skewness for value criterion and cost criterion*

tend to use more extreme values during prioritisation regardless of criterion. Others tend to be more modest and use the smaller values no matter which criterion. These subjects could perhaps have been satisfied with an ordinal scale technique.

This section has shown a possible way to calculate the skewness of the ratio distribution. This was used to investigate if certain subjects use the ratio scale potential more than others, which is confirmed by a slight correlation in our empirical data. It was also investigated if certain criteria tend to get a more skewed distribution. However, the average skewness is similar for both criteria within each data set.

Evaluation of ordinal scale cost-value diagram. The second research question regards the possibility to apply the cost-value approach when priorities are based on ranks. Usually, cost-value diagrams are used to visualise ratio scale priorities by percentages, see e.g. (Karlsson and Ryan, 1997). It may, however, be possible to draw similar diagrams with ordinal scale data, based on the ranks instead of the ratios, as described in Section 2.3. The cost-value diagram is used as decision-support when selecting the most appropriate set of requirements for a release. We want to investigate whether the same requirements would be selected for implementation when using ordinal data as when using ratio data in the cost-value diagram.

PROCEDURE. Two sets of cost-value diagrams were drawn based on our empirical data, one with ratio scale data and one with ranks. The ratio scale diagram corresponds to the example in Figure 2, option (a), i.e. the diagram is divided in three equally large areas. The ordinal scale diagram corresponds to the example in Figure 3. The ratio scale diagram was used as a baseline in the investigation. We wanted to see whether the ordinal scale diagram would point out the same requirements as the ratio scale diagram did. Therefore, we marked each requirement in the ordinal scale cost-value diagram with different symbols depending on where in the ratio diagram it appeared, see Figure 12. Each requirement with high contribution in the ratio scale diagram was marked with a circle; each requirement with medium contribution in the ratio scale diagram was marked with a square; and each requirement with low contribution in the ratio scale diagram was marked with a triangle. Thereby it was possible to compare the cost-value diagram based on ranks, with the original ratio cost-value diagram.

The ordinal scale cost-value diagram was divided into nine equally large areas, similar to the example in Figure 3. We assume that the three areas to the upper left, called A in Figure 3, should ideally contain high contributors according to the ratio scale and should be marked with a

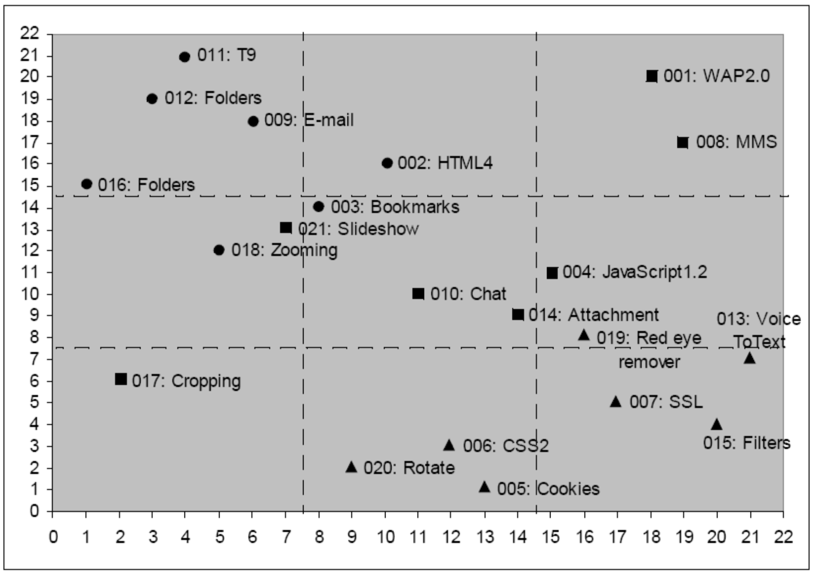


Figure 12. *Example of cost-value diagram based on ranks*

circle, if the ordinal scale was reflecting the ratio scale perfectly. Similarly, the three areas to the lower right, called C in Figure 3, should ideally contain low contributors according to the ratio scale and should be marked with a triangle. The middle areas, called B in Figure 3, should contain medium contributors, and be marked with a square.

Next we can calculate the level of agreement between the ordinal scale cost-value diagram and the original ratio scale diagram. The *Kappa value* (K) can be used to assess the agreement between a set of raters who have assigned a set of objects to one of several categories (Siegel and Castellan, 1988). In this case, the objects are the requirements and the categories are the three areas A, B and C. The two "raters" are the categorisation based on ranks and the categorisation based on ratios.

The Kappa value is calculated according to the following:

$$\textit{Eq 3.}\ \mathrm{K} = \frac{\mathrm{P(A)} - \mathrm{P(E)}}{1 - \mathrm{P(E)}}$$

where P(A) denotes the proportion of the times the raters agree and P(E) denotes the expected agreement that would be present by chance if all ratings were made randomly (Siegel and Castellan, 1988). Kappa values close to 1 represent very high agreement, while Kappa values close to, or below, 0 represent no agreement. The strength of agreement is classified by e.g. Landis and Koch (1977) as described in Table 19.

EXAMPLE. The cost-value diagram in Figure 12 is used as an example of how the requirements are distributed over the diagram when the ordinal scale is used. From this diagram it is possible to count the number of "correct" and "incorrect" requirements in each area.

For the example in Figure 12, it is visible that most requirements in area B are marked with a square, but some requirements have ended up in the wrong area. In this case, prioritising with an ordinal scale would miss

Table 19. Landis and Koch's Kappa statistics (Landis and Koch, 1977)

Kappa statistics	**Strength of agreement**
<0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost perfect

one high priority requirement, and would accidentally discard one medium priority requirement. However, most requirements actually end up in the correct area.

RESULTS. The tables below present the number of high (H), medium (M), and low (L) requirements in each of the three areas A, B, and C for each subject. The ideal categorisation, corresponding to the one in which the ranks reflect the ratios, would be that all requirements are in the correct area, i.e. area A contains high contributors, area B medium contributors and area C low contributors. The last row in the tables below presents the Kappa value, i.e. the measure of the agreement between the priorities based on ranks and the priorities based on ratio scale.

Table 20. Requirements in different areas of the cost-value diagram for data set 1

	A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	***Av***
Area A	3H	6H 2M	6H 1M	6H	7H 2M	5H 2M	7H 1M	6H 3M	
Area B	4H 7M 3L	1H 4M 1L	1H 5M	1H 6M 1L	2M 1L	2H 3M 2L	3M 1L	1H 2M 2L	
Area C	4L	1M 6L	1M 7L	1M 7L	1M 6L	3M 6L	3M 5L	3M 6L	
K	0.50	0.64	0.79	0.79	0.57	0.43	0.64	0.43	*0.60*

As can be seen in Table 20, subject A3 and A4 have the same Kappa value, as they have the same number of correctly placed requirements. Subject A6 and A8 have lower agreement because fewer requirements are correctly placed. The Kappa values for data set 1 represent moderate to substantial agreement.

Table 21. Requirements in different areas of the cost-value diagram for data set 2

	B1	**B2**	**B3**	**B4**	**B5**	**B6**	
Area A	7H 1M	4H 1M	6H	5H 1M	4H	5H 1M	
Area B	6M	3H 5M 3L	1H 6M	2H 6M 5L	3H 6M 5L	2H 4M 3L	
Area C	7L	1M 4L	1M 7L	5L	1M 2L	2M 4L	
K	0.93	0.43	0.86	0.64	0.36	0.43	

Table 21. (cont.) Requirements in different areas of the cost-value diagram for data set 2

	B7	**B8**	**B9**	**B10**	**B11**	**B12**	***Av***
Area A	6H 1M	6H 3M	6H	5H	3H 1M	7H 2M	
Area B	1H 5M 2L	1H 4M 1L	1H 6M	2H 7M 1L	4H 6M 2L	3M	
Area C	1M 5L	6L	1M 7L	6L	5L	2M 7L	
K	0.64	0.64	0.86	0.79	0.50	0.71	*0.65*

As Table 21 indicates, the Kappa values vary for data set 2 between fair agreement for subject B5 and almost perfect agreement for subject B1.

Table 22. Requirements in different areas of the cost-value diagram for data set 3

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	***Av***
Area A	2H	3H	3H	2H	3H	1H	3H	3H	
Area B	1H 2L	1M	2M	1H 2M 1L	2M	2H 2M 2L	2M	1M	
Area C	2M 1L	1M 3L	3L	2L	3L	1L	3L	1M 3L	
K	0.07	0.80	1	0.64	1	0.30	1	0.80	*0.72*

For data set 3, in Table 22, the Kappa values vary a lot between the subjects. Subject C1 has only slight agreement, subject C6 has fair agreement and subjects C3, C5, and C7 have perfect agreement between the ordinal and ratio scale diagrams.

Table 23. Requirements in different areas of the cost-value diagram for data set 4

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	***Av***
Area A	5H 1M	5H 2M	5H 2M	5H	5H 1M	5H 1M	4H 2M	4H	
Area B	2M 1L	2M 1L	3M	5M	3M	3M 1L	1H 4M	1H 5M 1L	
Area C	3M 4L	2M 4L	1M 5L	1M 5L	2M 5L	2M 4L	5L	1M 4L	
K	0.54	0.54	0.72	0.91	0.72	0.63	0.72	0.72	*0.69*

For data set 4, in Table 23, the Kappa value varies between 0.54 and 0.91, which corresponds to moderate, substantial or almost perfect agreement.

ANALYSIS. It seems as the areas A, B, and C are valid to use to distinguish the high, medium and low priority requirements, at least with the data sets in this investigation. The Kappa values correspond to, at least, fair agreement in all cases, except when only eight requirements were prioritised in data set 3. The small number of requirements makes each requirement affect the Kappa value more than in cases with a larger number of requirements. The Kappa values for data set 3 vary more than for the other data sets.

The average Kappa value for all data sets are 0.67, which reflects *substantial* agreement.

The conclusion drawn from this section is that it is possible to achieve similar decision-support with ordinal scale prioritisation techniques as with ratio scale prioritisation techniques. For most subjects in the investigation, only a few requirements are judged differently when basing the decision on the ordinal scale. Therefore, it would be possible to use the ordinal scale cost-value diagram as presented here when selecting requirements for a release. This is valuable as most ordinal prioritisation techniques are easier and faster to use than techniques with ratio scale results.

Opinions on scales. After the second experiment assignment, the fourteen subjects answered some questions regarding how they felt about the prioritisation technique and the scale. Approximately half of the subjects found it easy to decide which degree on the ratio scale to select, and half found it difficult. Motivations for finding it easy included that the scale was intuitive and the subjects felt sure about the domain. Subjects finding it difficult motivated it with lack of domain knowledge and some said that it was easy to choose which requirement that was more valuable, or expensive, but to decide how much was more difficult. Five of the subjects also stated that it would have been easier and faster to omit the ratio scale and only decide “more than” or “less than”. This would speak in favour of the ordinal scale. However, as some subjects pointed out, it would be more fault intense and yield less information. When asking the subjects whether they used the extreme values on the prioritisation scale, almost half of the subjects said that they did not. Their motivations include insecurity due to lack of domain knowledge, and that “you never know if another requirement is even better”. It is possible that when decision-makers are insecure about the domain, it may be sufficient to use the ordinal scale, since the extreme values of the ratio scale are not used.

4.3 Discussion

The main validity issue is the generalisability. As we have only investigated students and PhD students prioritising rather independent mobile phone requirements it is difficult to generalise to industrial cases. The skewness measure and the ordinal cost-value diagram need industrial validation in order to rely on the results with more certainty. The issue of participant and observer bias is reduced in this study as it is an archive analysis and neither the students nor the researchers planned to use the results for this purpose and could hence not affect them.

The data show a slight correlation (r=0.37, p=0.028) between the skewness for value and the skewness for cost. We need more data in order to determine whether the skewness for different criteria correlates. Similarly, it requires more investigation of the cost-value diagram in order to determine if it is valuable for decision-makers to use the ordinal scale. For most of our empirical data there appears to be a substantial agreement between the ratio scale cost-value diagram and the ordinal scale cost-value diagram. However, it requires use by decision-makers to see whether the ordinal scale diagram is sufficiently accurate. It shall also be noted that it is not necessarily the case that using an ordinal scale requirements prioritisation technique would give the same cost-value diagram as the one obtained by reducing the ratio scale data to ordinal scale data, which was done in this study. Further studies need to be made in order to determine if this transformation of data is an appropriate approximation of the results from ordinal scale prioritisation. An alternative approach to obtain comparable data on different scales would be to use data obtained from different techniques, one based on ordinal scale and one based on ratio scale. However, the difference in prioritisation techniques would then influence the result more than the difference in scales, which is the issue we try to investigate here. A comparison between ratio scale techniques and an ordinal scale technique is presented earlier in this part of the thesis.

The skewness may be used to compare different sets of data in order to determine which one that benefits more from the ratio scale. In our case, we compared the skewness between different subjects and between the criteria cost and value. There are other criteria as well that could be investigated similarly, such as risk, effort, business opportunity, etc. It would also be possible to compare prioritisation data from different domains, or different types of requirements, to see if certain domains benefit more from the ratio scale than others. The skewness is rather easy to calculate using a spread sheet, and it determines the skewness for each criterion separately. Instead of the linear baseline used for calculating the skewness, it would be possible to create a baseline based on the average distribution for the investigated subjects. In that manner, the skewness would then vary positively or negatively from the baseline distribution.

The cost-value diagram based on ranks can be used when the prioritisation technique results in ranks, which is the case when using e.g. the Planning game or Numeral assignment. For our investigation we chose to divide the diagram into nine equally large areas to simulate the high, medium, and low priority groups often resulting from e.g. Numeral assignment. However, in a real case, these groups might not be equally large and thus the nine areas might be of different sizes. Alternative ways to divide the cost-value diagram could result in other conclusions but this has not been investigated in this study.

4.4 Conclusions

This study has presented two ways to analyse and compare ordinal and ratio scale data: (1) Evaluation of skewness, and (2) Evaluation of ordinal scale cost-value diagram. It is important to be able to compare data from different scales since there is a trade-off between using a richer, but more time-consuming, scale and a less rich, but faster, scale when prioritising requirements.

We have presented a skewness measure that can be used to characterise the ratio scale data, see RQ1. The skewness can be used to compare the characteristics of the ratio scale for different subjects, criteria or types of requirements. Based on our empirical data we conclude that some subjects are more inclined to use the extreme values of the ratio scale than others. However, the skewness of the ratio scale does not seem to be affected by the different criteria, at least not cost and value. The practical implication of the presented results suggest that decision-makers can use the skewness measure in a pre-test before prioritisation to determine whether a ratio scale technique is worth the effort (due to high skewness) or if it is sufficient with an ordinal scale technique (due to low skewness).

If an ordinal scale prioritisation technique is used, it is valuable for decision-makers to be able to use the cost-value approach. Therefore, we have suggested how the cost-value diagram can be used to find the most valuable requirements, when priorities are on an ordinal scale, see RQ2. The empirical ratio scale data and the ranks of the same data were compared using cost-value diagrams. The decision regarding which requirements to select in the data sets used in this study would be rather similar basing the diagram on ranks instead of ratios.

This study has attempted to answer questions about how the difference between the scales can be calculated and illustrated. However, we cannot answer the even more interesting question about when the ratio information is worth the extra effort and when ranking of requirements may be enough. Below we have listed a number of ways to investigate this further:

- Conduct a case study where both ordinal and ratio scale prioritisation techniques are applied and evaluated by practitioners.
- Set up a controlled experiment where several different criteria are used, for example value, cost, effort, risk etc., to investigate if some criteria need the ratio scale more than others. Similarly, different types of requirements can undergo the same investigation.
- Perform an interview survey where several experienced practitioners answer questions regarding when the ratio scale is necessary and which characteristics that can be used to decide if the ratio scale should be used or not.
- Simulate priorities from different known distributions to compare with data from real ratio scale prioritisations to see which distribution that is most similar to the real ones – and to see if there are some distributions that can benefit more from the ratio scale than others.

5. Closing Remarks

Part II has presented two separate studies aimed at characterising and comparing requirements prioritisation techniques. Each study has its own discussion and conclusion sections since the studies have different research questions and goals that need to be discussed separately. In this final section, some concluding remarks are presented based on both papers.

The choice of requirements prioritisation technique depends on several different factors, such as time-consumption, ease of use, and accuracy. In addition, you need to consider what fits the development process used in the project. Furthermore, issues such as the cost and effort required for preparing the prioritisation session need to be regarded, as well as the cost and effort of introducing a certain prioritisation tool or technique. The first of the presented studies suggests that the Planning game seems feasible for requirements prioritisation because it is relatively fast and easy to use. It is also inexpensive compared to tool-based techniques, since it only requires paper and pen. Other sources confirm that the requirements prioritisation practice in many organisations is based on grouping requirements, rather than using more rigorous techniques (Lehtola and Kauppinen, 2006). Therefore, introduction of Planning game might be efficient and successful. Lehtola and Kauppinen (2006) also discovered that some users of a ratio scale technique found it difficult to determine how much more important one requirement is than another. In addition, in the second study in Part II, some subjects did not use the extreme values on the ratio scale. Such subjects could perhaps have been satisfied with an ordinal scale technique. The second study also implies that it is possible to use the cost-value approach on ordinal scale data obtained using the Planning game. All these issues speak in favour of the Planning game technique.

However, ratio scale techniques are beneficial when we need to regard the relative differences between requirements. These differences may be valuable to consider, since it can turn out to that a few requirements account for a large part of the total value. Ratio scale techniques also yield more informative results since the difference in priority between requirements comes out clearer. The skewness measure presented in the second study can be used to determine the cases when ratio scale techniques are beneficial due to the large differences in priority between requirements. Therefore, the skewness measure is valuable when determining which technique to choose. The measure can be used in retrospect to measure the skewness on historical data or it can be used in a pre-study to find situations when the ratio scale is beneficial.

Appendix

Table A. Experiment 1 using counter-balancing design

Subject	Nbr of Features	Tech 1	Tech 2	Criterion 1	Criterion 2
1	8	PWC	PG	Price	Value
2	8	PWC	PG	Price	Value
3	16	PWC	PG	Price	Value
4	16	PWC	PG	Price	Value
5	8	PWC	PG	Value	Price
6	8	PWC	PG	Value	Price
7	16	PWC	PG	Value	Price
8	16	PWC	PG	Value	Price
9	8	PG	PWC	Price	Value
10	8	PG	PWC	Price	Value
11	16	PG	PWC	Price	Value
12	16	PG	PWC	Price	Value
13	8	PG	PWC	Value	Price
14	8	PG	PWC	Value	Price
15	16	PG	PWC	Value	Price
16	16	PG	PWC	Value	Price

Table B. Experiment 2 using counter-balancing design

Subject	Occasion	Tech 1	Tech 2	Criterion 1	Criterion 2
1	PM	TPWC	PG	Value	Price
2	PM	TPWC	PG	Value	Price
3	PM	TPWC	PG	Value	Price
4	PM	TPWC	PG	Value	Price
5	EV	TPWC	PG	Value	Price
6	EV	TPWC	PG	Value	Price
7	EV	TPWC	PG	Value	Price
8	PM	TPWC	PG	Price	Value
9	PM	TPWC	PG	Price	Value
10	PM	TPWC	PG	Price	Value
11	PM	TPWC	PG	Price	Value
12	PM	TPWC	PG	Price	Value
13	EV	TPWC	PG	Price	Value
14	EV	TPWC	PG	Price	Value
15	PM	PG	TPWC	Value	Price
16	PM	PG	TPWC	Value	Price
17	PM	PG	TPWC	Value	Price
18	PM	PG	TPWC	Value	Price
19	EV	PG	TPWC	Value	Price
20	EV	PG	TPWC	Value	Price
21	EV	PG	TPWC	Value	Price
22	PM	PG	TPWC	Price	Value
23	PM	PG	TPWC	Price	Value
24	PM	PG	TPWC	Price	Value
25	PM	PG	TPWC	Price	Value
26	PM	PG	TPWC	Price	Value
27	PM	PG	TPWC	Price	Value
28	PM	PG	TPWC	Price	Value
29	EV	PG	TPWC	Price	Value
30	EV	PG	TPWC	Price	Value

Table C. Requirements prioritised in the experiments

Feature	Selected for 8 features
Alarm	x
Bluetooth	
Calculator	
Calendar	x
Call alert creation	
Colorscreen	x
Games	x
IR	
MMS	
Notebook	x
Phonebook	
SMS	
Timer	x
WAP	x
Vibrating call alert	x
Voice control	

Table D. Requirements prioritised in the laboratory sessions, data set 1 and 2

Requirement	
001: WAP 2.0	012: Folders
002: HTML4	013: VoiceToText
003: Bookmarks	014: Attachment
004: JavaScript1.2	015: Filters
005: Cookies	016: Folders
006: CSS2	017: Cropping
007: SSL	018: Zooming
008: MMS	019: Red eye remover
009: E-mail	020: Rotate
010: Chat	021: Slideshow
011: T9	

PART III: RETROSPECTIVE ANALYSIS FOR RELEASE PLANNING DECISIONS

III

Paper 4. Case Studies in Process Improvement through Retrospective Analysis of Release Planning Decisions.

Lena Karlsson, Björn Regnell, Thomas Thelin

Accepted for publication in International Journal of Software Engineering and Knowledge Engineering: Special Issue on Requirements Engineering Decision Support, December 2006.

Paper 5. Retrospective Analysis of Release Planning Decisions in a Product Line Environment - A Case Study.

Lena Karlsson, Björn Regnell

Submitted, 2006.

Paper 6. Introducing Tool Support for Retrospective Analysis of Release Planning Decisions.

Lena Karlsson, Björn Regnell

Proceedings of the 7th International Conference on Product Focused Software Process Improvement (PROFES'06), Amsterdam, the Netherlands, June 2006.

Abstract

The process of selecting requirements for a release of a software product is challenging as the decision-making is based on uncertain predictions of customer value and development cost. This part of the thesis presents a method aimed at supporting software product development organisations in the identification of process improvement proposals to increase requirements selection quality. The method is based on an in-depth analysis of requirements selection decision outcomes after the release has been launched to the users. The method is validated in three separate case studies involving real requirements and industrial practitioners. The conclusions from the case studies include that the method seems valuable in situations with complex release planning decisions. It appears essential that participants with different viewpoints attend the root cause discussion. Requirements interdependencies seem to play a big role in release planning decision-making. Successful projects can also be a source of learning. In addition, it is valuable to combine the PARSEQ method with actual project data collected in e.g. a knowledge management effort.

1. Introduction

This part of the thesis presents a method for identifying improvement areas in the release planning process through conducting retrospective analysis. The method is called PARSEQ (Post-release Analysis of Requirements SElection Quality). The method is based on retrospective examination of release planning decision-making, at a time when the consequences of requirements selection decisions are visible. Release planning decisions determine in which release of the product different requirements should be included. These decisions may affect the success of a product (Carlshamre, 2002). Release planning was discovered to be an important issue in Paper 1. Interviewees stated for example that the release plan is dependent on accurate time-estimates since the estimates affect how many requirements that can be selected for each release. Therefore, we view release planning as an important but difficult activity, which need support for improvement.

Given issues such as uncertain estimates of requirements user value and cost of development, it can be assumed that some requirements selection decisions are non-optimal. This in turn may lead to software releases with a set of features that are not competitive or that do not satisfy user expectations. Only afterwards, when the outcome of the development effort and user value is apparent, is it possible to tell with more certainty which decisions were correct and which were not. By looking at the decision outcomes in retrospect, organisations can gain valuable knowledge on how to improve the requirements selection process and increase the chance of product success. Retrospective analysis may help focus on the most acute and valuable improvements for the release planning process.

This part of the thesis includes an integrated presentation of results from three case studies (Paper 4 and 5). The contribution includes conclusions on the range of applicability of retrospective analysis for release planning decisions. The PARSEQ method needs to be adaptable to projects of different size, maturity and level of agility. Each step of the method can take different shapes, depending on the characteristics of the project of interest. In particular, a discussion of light-weight versus advanced versions of components within the methodology is provided.

In the first case study, the method was applied at a company developing a software product for an open market. The company had regular releases and used a requirements management (RM) tool called

Focal Point (Telelogic, 2006) when planning their releases. Several improvement suggestions for the release planning activity were found during the case study, e.g. enhancing the overall picture of related requirements, paying more attention to the elicitation of usability requirements and improving estimates of implementation costs.

In the second case study we applied the method with a more agile approach. The investigated case used an agile development process and the software system was developed in an in-house project, i.e. both users and developers of the project were from within the company. The agile application of PARSEQ seems promising and the release planning activity in the investigated project was found successful. The participants concluded that the iterative development, with flexible release plans, was beneficial. The project type may also have influenced the results since users of in-house projects have similar requirements, and they can express their requirements to the developers who are located close by.

In the third case study we investigated product line development (Clements and Northrop, 2002) of an embedded consumer product. In addition, a prototype tool developed to support the method was evaluated, see Paper 6. The product line situation entails special challenges since it requires the products to be adapted for reuse and several products are alive at the same time. In addition, embedded products often have more difficult dependencies since both software and hardware aspects need to be considered. The PARSEQ method resulted in several process improvement suggestions regarding the process, market focus, and development issues. The prototype tool was developed based on the experiences from the first two case studies. It seems to support the PARSEQ analysis by increasing efficiency and visualisation potential of release planning problems through charts and diagrams, as well as decreasing preparation and manual handling of requirements through import and export possibilities.

The three main issues that differ between the case studies are the project type, the product type, and the development approach. The first case study presents a market-driven project using incremental development for a software product. The second case study presents an in-house project using an agile development approach for a software product. The third case study presents a market-driven project using a product line development approach for an embedded product. The specific project types make it difficult to generalise to other organisational settings with different customer-supplier relationships. Qualitative research designs, such as case studies, do not attempt to generalise results beyond the case under study. Instead, the intent is to focus on the credibility and trustworthiness of the results (Robson, 2002). In addition, qualitative research aims to investigate as different cases as possible in order to reveal possibilities and limitations of the investigated method. The three reported cases are different and complement each other. The results were fed back to the participants who found the improvement proposals relevant. The relevance of the results supports the idea that retrospective analysis can be a valuable means for process improvement, although the actual benefit from the process improvement proposals might not be visible in a short-term perspective.

The investigated research questions are:

RQ1. Do the participants find the PARSEQ analysis valuable?

RQ2. Do the results of the PARSEQ analysis differ depending on the project type, product type and development approach?

RQ3. What are the lessons learned about the adaptations of the PARSEQ method to the cases?

RQ4. Can the PARSEQ prototype tool make the PARSEQ analysis more efficient and illustrative?

Part III is structured as follows. Section 2 presents related work, including retrospective analysis, release planning, and software product line engineering. Section 3 describes the PARSEQ method and its five steps, and Section 4 describes the prototype tool used in the third case study. Section 5 presents the three case studies and the results from applying the PARSEQ method. Section 6 discusses the findings in the case studies and Section 7 concludes with some lessons learned and ideas for future work.

2. Related Work

This section describes some related work in the areas of retrospective analysis, release planning, and software product line engineering.

2.1 Retrospective Analysis

In (Kerth, 2001), retrospective analysis is acknowledged as one of the most important means for software process improvement. This activity has many different names - process review, post mortem analysis, project postmortem, etc., but the idea is the same: by looking back and learning from the past it is possible to improve the future. Retrospective analysis is not new, nor is it only applicable to software projects. However, physical products often have built-in feedback, making it more evident whether or not you have succeeded. In the world of software development, the product is more or less invisible and this invisibility heightens the need for feedback on product and project success (Kerth, 2001).

The project retrospective is considered an excellent method for Knowledge Management, since it captures experience and improvement suggestions from completed projects (Birk et al., 2002; Rus and Lindvall, 2002). One of the first papers to present a defined process recommends a five step approach that includes designing a project survey, collecting objective project metrics, conducting a debriefing meeting followed by a "project history day", and finally publishing the results (Collier et al., 1996).

Later on, a more lightweight retrospective review was proposed (Birk et al., 2002). It is based on a focused brainstorm on what happened in the project and a technique called the "KJ Method". The participants get three to five post-it notes and are asked to write down one "issue" on each. The post-its are attached to a whiteboard and are then grouped, discussed and prioritised. Then the issues are analysed in a Fishbone diagram to find the causes for positive and negative experiences.

In (Nolan, 1999), the author states that the most effective way to improve is by learning from success. Rather than studying the unsuccessful projects or looking for external answers, you can choose your most successful projects and learn from them. While traditional approaches to process improvement often focus strongly on improving through perpetual refinement, it is suggested that learning from success is a more effective and efficient scheme. The author states: "If you believe that failure is no accident, then you must also believe that success is no accident".

Retrospective reviews are often discussed in a project management context (Cleland, 1995; Ulrich and Eppinger, 2000). An evaluation of the project's performance after it has been completed is useful both for personal and organisational improvement and can be conducted as an open discussion of the strengths and weaknesses of the project plan and execution. Much can be learned about organisational efficiency and effectiveness from this kind of evaluation, as it offers an insight into the success or failure of a project. The lessons learned can be used when planning future projects to improve project performance and to prevent mistakes.

Continuous process improvement is important in mature software development. In particular, requirements engineering is pointed out as a critical improvement area in a maturing organisation (Paulk et al., 1995). A recent process improvement study based on analysis of defects in present products is reported in (Lauesen and Vinter, 2001).

As the retrospective analysis has proven to be an effective way of learning and improving, the practice has also been adopted by the agile community. Agile methodologies share an iterative and incremental way of working. This is also reflected in how they recommend performing retrospective analysis. "People have to develop the habit of looking for process refactorings just like they look for code refactorings" (Collins and Miller, 2001). The agile community takes the project retrospective one step further and suggests retrospectives to be performed after each iteration. The aim is to reach a more continuous improvement, so that problems can be found and corrected before the product is released.

2.2 Release Planning

Deciding what to develop, and when, is a complex task. The common trade-off between customer value and implementation cost must be taken into consideration but other factors may also be important such as available resources, benefit for the company, competitors' product plans, logical implementation order, and cost if *not* implemented (Lehtola et al., 2004). Developing products in an incremental manner provides the opportunity of releasing the most important functionality first instead of delivering a monolithic system after a long development time. Consequently, customers receive part of the system early on and are more likely to provide feedback on it. It is also easier to estimate the cost and schedule for each delivery, as each release is smaller. An incremental approach also allows for a better reaction to changes or additions to requirements (Greer and Ruhe, 2004).

The incremental agile methodologies have gained interest during the last years as they provide a flexible alternative to the traditional development approaches. The most well-known approach may be Extreme Programming (XP) (Beck, 2005). XP involves twelve practices such as Pair programming, Continuous integration, Refactoring, and the Planning game. The Planning game is a procedure used to determine the scope of the next release by combining business priorities, e.g. value to the user or customer, and technical estimates, e.g. time, risk, and resource estimates.

Delivering software incrementally necessitates a process of analysing requirements and assigning them to increments. The release planning task is normally preceded by requirements prioritisation and resource estimation (Carlshamre, 2002). Several different techniques for requirements prioritisation exist, see e.g. (Berander and Andrews, 2005; Karlsson et al., 1998). However, at the time of prioritisation it is difficult to be fully aware of the context and circumstances present when the release is launched. The issue of resource estimation has been debated within software engineering since its origin and is described in e.g. the COCOMO model (Fenton and Pfleeger, 1997). Furthermore, requirements are often subject to interdependencies pertinent to release planning, which radically increases the complexity of the selection task. In (Carlshamre et al., 2001), it is discovered that customer-specific development tends to include more functionality-related dependencies, whereas market-driven product development have an emphasis on value-related dependencies.

One technique for release planning is the Planning game described above. Another one is the EVOLVE approach (Greer and Ruhe, 2004), which takes multiple aspects into consideration when assigning requirements to increments, e.g. different stakeholder perspectives, effort constraints, requirements interdependencies and changing requirements. The approach uses a genetic algorithm to derive potential release plans within pre-defined constraints.

2.3 Software Product Line Engineering for Embedded Products

According to (Clements and Northrop, 2002) a *software product line* (SPL), or a software product family, is "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way". The core assets from which the different products are developed can also be called a *product platform* and include the architecture, reusable components, domain models, requirements statements, documentation, schedules, test plans, process descriptions, etc.

The concept of product platforms is not new but has been used in, e.g., the automotive industry for decades. In that case, the platform is a piece of hardware on top of which other hardware and software is built. This kind of *embedded software* appears in an increasing number of products in addition to cars: machine tools, home appliances and even toys. Production costs can be lowered significantly by using the same mechanical configuration for various models of a product line but shipping them with different controlling software. Embedded software is rapidly becoming a key differentiator between competitive products (Miller, 1998).

Embedded systems have additional aspects to consider, not apparent in traditional software development, e.g., performance, cost, emissions, power consumption, and weight (Sangiovanni-Vincentelli and Martin, 2001). Embedded systems have more difficult trade-offs than traditional software development, which is discussed by Graaf et al. (2003) in their interview study of embedded software state-of-the-practice. They found that hardware development primarily dominated system development because of longer lead times and logistical dependencies on external suppliers. Consequently, software development started when hardware development was already at a stage where changes would be expensive. Hardware properties then narrowed the solution space for software development. In some cases software acted as an integrator, i.e., problems that should have been solved in the hardware domain were solved in the software domain (Graaf et al., 2003).

The product platform is required to be adapted for *reuse*, which puts special demands on nonfunctional requirements, such as real-time aspects, performance and reliability (Graaf et al., 2003). In addition, release planning is more complex since we need to consider releases at several different levels. Each product in the product line have different releases, and the platform itself is also subject to release planning. Thus, several products in the product line are active at the same time as opposed to the ordinary release planning situation where a product update means that the prior release does not need further consideration.

3. The PARSEQ Method

Retrospective evaluation of software release planning may give valuable input to the identification of process improvement proposals. In particular, post-release analysis of the consequences of previous decision-making may be a valuable source of information when finding ways to improve the requirements selection process.

The PARSEQ method is based on a systematic analysis of candidate requirements from previous releases. By identifying and analysing a set of root causes to suspected incorrect requirements selection decisions, it may be possible to find relevant improvements that are important when trying to increase the specific organisation's ability to plan successful software releases.

In order to use the PARSEQ method, the following foundation practices are required:

- Multiple releases of the product and requirements from earlier releases are saved in a repository.
- Data for each requirement stating in which release it is implemented, or if the requirement has been postponed or excluded.
- Methods for estimating each requirements' cost and value.
- Employees who have decision-making experience from prior releases.
- A facilitator with experience in performing retrospective analysis.

PARSEQ is divided into five steps: requirements sampling, re-estimation of cost and value, root cause analysis, elicitation of improvements, and prioritisation of improvements, as shown in Figure 1. Each of the steps can be adapted to the particulars of a case study. For example, there are different approaches to requirements sampling that can be selected for the first step. There are also several different techniques for requirements prioritisation that can be used in the second step.

The method uses a requirements repository as input and assumes that information is available regarding when a requirement is issued and in which release a requirement is implemented. The output of the method is a list of prioritised process improvement proposals and a plan for improvement implementation. Each step in PARSEQ is subsequently described in more detail.

Step 1: Requirements sampling.

The main input to the retrospective analysis is a list of requirements that were candidates for previous releases of the investigated product. The product should have been in operation long enough to allow for an assessment of the current user value of its implemented requirements.

The purpose of the sampling is to compose a reasonably small but representative sub-set of requirements, since the complete repository may include several hundreds of requirements. Requirements sampling makes the retrospective analysis more time-efficient so that it can take place during a workshop. The sample should include requirements that were selected for implementation in one of the releases as well as postponed or excluded requirements. The requirement set is thereby useful for the analysis as it consists of typical examples of release planning decisions.

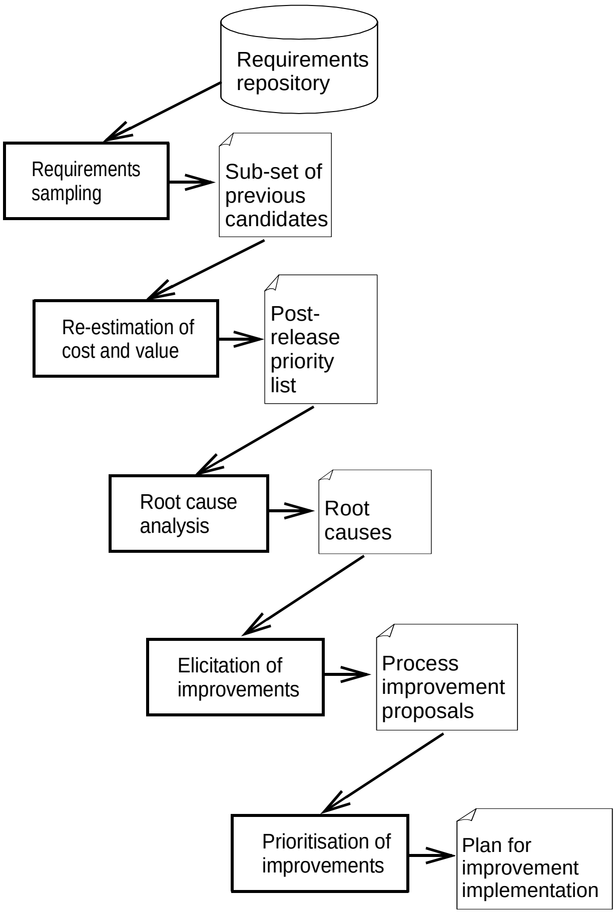

Requirements repository
Requirements sampling
Sub-set of previous candidates
Re-estimation of cost and value
Post-release priority list
Root cause analysis
Root causes
Elicitation of improvements
Process improvement proposals
Prioritisation of improvements
Plan for improvement implementation

Figure 1. *An outline of the activities and products of the PARSEQ method*

The requirements sampling can be performed with different focus, such as concentrating on a special market segment or on a difficult part of the product or on particularly difficult decisions. However, if the sample is supposed to represent the whole product and its users, the sample should be as comprehensive as possible. If a random sample is used, the following types of requirements may be excluded:

- Very similar requirements, since they do not extend the sample.
- Requirements dated several releases ago, as they may have evolved out of scope.
- Requirements dated recently, since they might not yet be analysed.
- Requirements estimated to have a very long or very short implementation time, as they are atypical and likely to be split or joined.

The output from the requirements sampling is a reasonable amount of requirements, large enough to be representative, yet small enough to allow the following steps of PARSEQ to be completed within a reasonable time.

Step 2: Re-estimation of cost and value

The requirement sample is input to the next step of PARSEQ, where a re-estimation of current user value and actual development cost is made in order to find suspected inappropriate decisions that can be further analysed. Since the investigated product releases have been in operation for a while, a new assessment can be made, which applies the knowledge gained after the releases were launched. Presumably this results in more accurate priorities. The re-estimation determines how the organisation would have decided, i.e. which requirements would have been selected, if it knew then what it knows now. With today's knowledge about user expectations and development costs, a different set of requirements may have been selected for implementation in the different releases. If this is not the case, either the organisation has not learned more about release planning since the releases were launched, or the release planning decisions were accurate.

The implemented requirements have a known development cost (assuming that actual implementation effort is measured for each requirement), but postponed or rejected requirements need to be re-estimated based on architectural decisions and the knowledge gained from the actual design of the subsequent releases.

By using, for example, a cost-value prioritisation approach, it is possible to see the trade-off between the value to the users and the cost of development in a so called cost-value diagram (Karlsson and Ryan, 1997), or in a bar chart (Telelogic, 2006). These illustrations point out the requirements with high value and low cost (they should be implemented early), as well as the requirements with low value and high cost (they should be implemented late or perhaps not at all).

The purpose of the re-estimation is to apply the knowledge that has been gained since the product was released to discover decisions that would be made differently today. The discrepancies between the decisions made during release planning and during post-release prioritisation are noted and used in the root cause analysis. The output of this step is thus a list of requirements that were given a high post-release priority but were implemented late or not at all, as well as requirements that were given a low post-release priority but were implemented in an early release.

Step 3: Root cause analysis.

The purpose of the root cause analysis is to understand on what grounds release-planning decisions were made. By discussing prior release planning decisions, and determining root causes of problematic ones, it may be possible to determine what went wrong and recommend how to do it better next time.

The output of the re-estimation, i.e. the discrepancies between the post-release prioritisation and what was actually selected for implementation in the different releases, is analysed in order to find root causes to the suspected inappropriate decisions. This analysis is based on a discussion with persons involved in the requirements selection process. Questions such as the following can be used to stimulate the discussion and provoke insights into the reasons behind the decisions:

- Why was the decision made?
- Based on what facts was the decision made?
- What has changed since the decision was made?
- When was the decision made?
- Was it a correct or incorrect decision?

Guided by these questions, categories of decision root causes are developed. Each requirement found to be implemented either too early or too late is mapped to one or several of these categories. This mapping of requirements to root cause categories is the main output of this step together with the insights gained from retrospective reflection.

Step 4: Elicitation of improvements.
The outcome of the root cause analysis is used to facilitate the elicitation of improvement proposals. The objective of this step of PARSEQ is to arrive at a relevant list of high-priority areas of improvement. The intention is to base the discussion on strengths and weaknesses of the requirements selection process and to identify changes to current practice that can be realised. The following questions can assist to keep focus on improvement possibilities:

- How could we have improved the decision-making?
- What would have been needed to make better decisions?
- Which changes to the current practices can be made to improve requirements selection decisions in the future?

If this step results in multiple improvement proposals, it may be necessary to prioritise between improvement suggestions.
Step 5: Prioritisation of improvements.
The input to the fifth and last step of the PARSEQ method is a list of process improvement proposals for the release planning process. In order to decide which improvements to implement, it may be necessary to prioritise between the suggested improvements so that the most cost-effective and important ones can be dealt with first. The prioritisation can be performed in a similar manner as in step 2, i.e. based on cost and value. The results of PARSEQ can then be used in a process improvement programme where process changes are designed, introduced and evaluated. These activities are, however, beyond the scope of the PARSEQ method.

4. The PARSEQ Tool-Support

In order to support the retrospective analysis method a prototype tool was developed (Paper 6). The experiences from conducting the two first PARSEQ case studies (see Section 5.1 and 5.2) were used when developing the tool, which handles all steps from the import of a sample of requirements to the export of process improvement proposals. The tool consists of a number of different windows guiding the user through the PARSEQ process.

Step 1: Requirements sampling.

The tool can import a list of requirements from MS Excel^{TM}. It requires the imported list to be in a certain format with columns representing requirement number, requirement description and release number. Therefore, the original requirements repository may need to be altered before importing it. The tool supports manual entering of requirements as well as editing of imported requirements. The tool cannot manage random sampling, only manual selection of requirements from the imported list. Therefore, it is more efficient to conduct the sampling prior to the PARSEQ session, so that manual selection does not have to be performed.

Step 2: Re-estimation of cost and value.

When requirements have been imported into the tool along with each requirement's release number and description, the second step can be performed. The re-estimation of cost and value is performed by using one of the three available requirements prioritisation techniques. We chose to include the two techniques that were used effectively in the first two case studies, i.e. the Pair-wise comparisons (Saaty, 1980) and the Planning game (Beck, 2005), and in addition we included the $100 technique (Leffingwell and Widrig, 2000), as it has been used successfully by the researchers before (Regnell et al., 2001).

At this step it is also necessary to select the criteria on which to base the prioritisation. Pre-defined criteria include cost, value, and risk. It is also possible to enter two criteria of your own choice. It is essential to choose one criteria to maximise, e.g. value, and one to minimise, e.g. costs.

The PWC requires the user to perform pair-wise comparisons between all possible pairs of requirements. As the number of comparisons increases drastically with the number of requirements, this is very time-consuming for large amounts of requirements. However, there are different algorithms to reduce the number of comparisons, for example the Incomplete Pair-wise Comparisons (IPC) (Harker, 1987). The tool implementation of the PWC prioritisation technique was inspired by the IPC. In the tool, it is possible to stop before all pairs are compared and receive an approximate value. This reduces the necessary comparison effort, but also the trustworthiness of the result. The same process is performed for both selected criteria.

The Planning game helps the decision-maker to rank requirements by first assigning each requirement to the high, medium or low box, and then arrange them in falling order within each box. This is performed for both selected criteria, for example cost and value, before continuing.

The $100 technique prioritises the requirements by providing each requirement with a share of a total budget of 100 dollars. This gives each requirement a percentage of significance according to the currently used criteria. A cell at the bottom keeps track of the total amount spent so far.

Step 3: Root cause analysis.

In the root cause analysis, the requirements identified as implemented too early or too late are analysed. The support provided by the tool for this step is divided into two windows, the Graph window and the Root cause matrix. In the Graph window, the results from the re-prioritisation are displayed in a cost-value diagram. Each requirement's position is shown with an icon: a "+" or one or more circles, to tell releases apart. The requirements with a "+"-sign have no release number assigned, representing that the requirements are not yet developed. The number of circles are decided in alphabetical or numerical order, for instance if you have releases 1, 2 and 3 they would have one, two and three circles respectively.

After discussing the root causes of making incorrect release planning decisions for a certain requirement, the requirement can be added to the Root cause matrix. In the matrix, selected requirements end up in columns, and root causes can be entered in each row. By marking an "X" in the appropriate cell, it is possible to assign root causes to requirements.

In the cost-value diagram it is possible to mark requirements dependencies by drawing a line between related requirements and writing a note about the relation.

Step 4: Elicitation of improvements.

At this point it is desirable to discuss the root causes and reasons for making incorrect decisions. Possible improvements to manage the incorrect decisions in the future can be entered next to the root causes in the root cause matrix. This is evidently a very important step of the method and it requires intense discussion between decision-makers.

When improvement proposals have been extracted, it is possible to export the results back to MS Excel^{TM}. Both the root cause matrix and the cost-value diagrams can be exported and used for presentation purposes in an improvement programme.

Step 5: Prioritisation of improvements.

Finally, it is possible to prioritise the improvement proposals based on e.g. the importance of putting the improvement in operation and the cost of doing so. In this manner the cost-value approach is used again. This is achieved by performing the first steps of PARSEQ again: importing the improvement proposals in the same manner as when requirements were imported, and then re-prioritising the improvements using one of the prioritisation techniques. The resulting cost-value diagram can indicate the most important, yet cost-effective, improvements to implement first. The root cause matrix can be used to enter reasons for implementing a certain improvement and plans on how this can be done. It would also be possible to add notes about dependencies between improvement proposals if, for example, conflicting proposals are found. The improvements, the cost-value diagram and the root cause matrix can again be exported to MS Excel^{TM}.

5. The PARSEQ Method in Case Studies

Three consecutive case studies have been conducted to examine the PARSEQ method in practice. The first case study took place at a small software product developer. The development followed an incremental approach with regular releases every 6 months. Users and customers were external as the product was sold on an open market. We used a commercial RM tool in the second step of the method. The second case was an internal project at a medium-sized company developing embedded software products. The project used an agile development approach, and as we wanted to look at a more agile alternative to the RM tool, the Planning game was selected for re-estimation of cost and value. The third case study took place at a large company developing embedded consumer products. They use a product line approach to development and have several sub-contractors. We used the Planning game for re-estimation of cost and value. The characteristics of the case study organisations are described in Table 1.

Due to lack of time and the fact that the cases did not result in numerous process improvement suggestions, it was decided to omit the fifth and final step of the method: Prioritisation of improvements. Therefore, this final step has not yet been evaluated empirically.

Table 1. Comparison between the three studied cases

	Case A	**Case B**	**Case C**
Project type	Market-driven development	In-house	Market-driven development
Product type	Software	Software	Embedded
User base	Multiple, diverse views	Few, similar views	Multiple, diverse views
User location	Outside organisation	Within organisation	Outside organisation
Development approach	Incremental	Agile	Product-line
Organisation size	Small	Medium	Large

5.1 Case Study A: Market-Driven Software Product

PARSEQ was tried out in a first case study to investigate its feasibility and gain more knowledge for future research on retrospective analysis of requirements selection as a vehicle for process improvement. In the first section, the case study site and context is described as well as the RM tool used in the study. Next, the realisation of the PARSEQ method is described, i.e. how each step of the method was carried out in the case study. Finally, the results from the case study are reported, including a number of improvement proposals.

5.1.1 Background

The case study site is a small organisation developing stand alone software packages. The organisation stores the requirements for the software package in a database that contains already implemented requirements as well as suggestions for new requirements. Each requirement is tagged with its level of refinement. Examples of states include New, Accepted for prioritisation, Accepted for implementation and Done, see Figure 2. When a requirement for some reason is not appropriate for the package, its state is set to Rejected.

To analyse the requirements in the database a commercial tool for product management and requirements management, called Focal Point

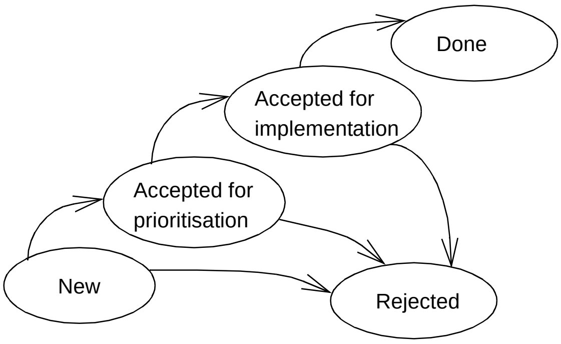


Figure 2. *A simplified version of the requirement state model in the database*

(Telelogic, 2006), was applied. The tool has capabilities for eliciting, reviewing, structuring, and prioritising requirements as well as for planning optimal releases that maximise the value for the most important customers in relation to development time and available resources. One prioritisation method in Focal Point is pair-wise comparisons, which results in priorities on a ratio scale (Fenton and Pfleeger, 1997). The tool provides solutions for reducing the number of comparisons and motivating the priorities. The tool aids in visualising the decisions in a number of different chart types. Due to redundancy of the pair-wise comparisons, the tool also includes a consistency check (Karlsson and Ryan, 1997) that describes the amount of judgement errors that are made during the prioritisation.

5.1.2 Conduct of Study

The participating organisation was given the opportunity to use PARSEQ to reflect on a set of decisions made during prior releases. The case study was executed during a one-day session, with approximately 5 hours of efficient work. Two researchers acted as facilitators during the session.

Step 1: Requirements sampling.

We analysed a release that was launched 12 months earlier, as it was of special interest to the company. Subsequently that release is called the reference release. Since then, another release had been launched and yet another one was planned to be released in the near future.

The requirements database contained more than 1000 requirements that were issued before the reference release and implemented in either that release or postponed to one of the following ones. Of these

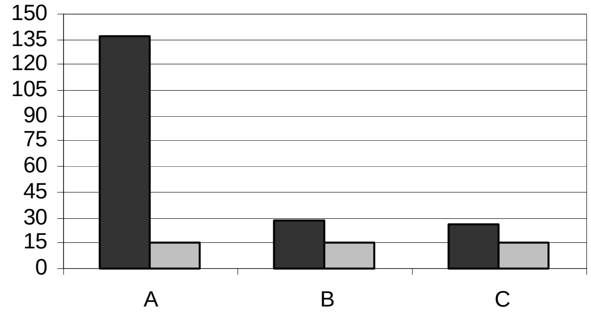


Figure 3. *Number of implemented requirements (dark grey) in each release compared to the sample (light grey)*

requirements, 45 was considered a reasonable number to extract. The requirements were equally allocated over the three releases: A, B and C, i.e. 15 were implemented in reference release A, 15 in release B and another 15 were planned for release C.

Note that the releases were not equally large in terms of number of requirements, i.e. the samples are not representative. The 15 requirements from release A were selected from 137 requirements, while the releases B and C only consisted of 28 and 26 requirements, respectively, as shown in Figure 3.

The requirements were selected randomly from a range where the ones estimated as having a very high, or very low, development effort had been removed, since they are not considered as representative. Very similar requirements had also been excluded to get the broadest possible sample.

All market changes, architectural decisions and new knowledge gained during the 12 months between the reference release A and release C were considered. The selected requirements were all in the states Done or Accepted for implementation; no rejected or postponed requirements were considered in the analysis. The requirements sampling took approximately one hour and was performed by a developer before the session.

Step 2: Re-estimation of cost and value.

The re-estimation was performed to find out which requirements the decision-makers would have selected for release A if they knew then what they know now. With the knowledge gained since the reference release was planned, it is possible that a different set of requirements would have been selected. However, it is important to note that one additional requirement in the release would imply that another one has to be removed, in order to keep the budget and deadline.

The market value was estimated using pair wise comparisons and the cost was estimated in number of hours, based on expert judgement. The following question was used in the pair-wise comparison of the candidate requirements of the reference release: "Which of the requirements would, from a market perspective, have been the best choice for release A?". This question focused on the retrospective nature of the estimation. Thus, the assessment concerned the market value given what is known today about the reference release.

The 45 requirements were re-estimated by using the Focal Point tool and pair-wise comparisons to prioritise them based on the selected question. The prioritisation was performed by a marketing person, who has good knowledge of customer demands, guided by a developer, and was attended by two researchers. Both the marketing person and developer had performed the original estimations, otherwise the results may be biased by differences in personal opinions rather than a desired effect of changes in priorities over time. When uncertainties or disagreements of a comparison occurred, the issue was briefly discussed to come to an agreement. The consistency check showed that the prioritisation was carefully performed and only two comparisons had to be revised and changed.

The total time for prioritisation was just over one hour, during which 70 comparisons were made. The short time is due to the tool, which reduces the number of comparisons to less than *2n* and points out inconsistencies among comparisons (Karlsson an Ryan, 1997; Harker, 1987). Otherwise, the number of comparisons would have been *n(n-1)/2*, which in this case equals 990.

The development cost of the requirements that were actually implemented was known, while the development cost of the requirements that are planned for a coming release had to be re-estimated. However, it was decided to use the available cost estimations, since they had recently been reviewed and updated.

A bar chart was created in the Focal Point tool to visualise and facilitate analysis of the decisions, see Figure 4. The grey bars illustrate the requirements implemented in release A, and the white bars represent requirements implemented in release B or planned for release C. The area to the right represents the re-estimated relative value, and the area to the left represents the re-estimated relative cost.

The prioritisation in the tool is performed on a ratio scale, using a process similar to the Analytical Hierarchy Process (AHP) (Karlsson et al., 1997; Saaty, 1980). Therefore, it is possible to subtract the cost from the value, getting a resulting priority, which is marked by the black arrows in the bar chart. The bars are sorted based on their resulting priority from top down. Thus, the bar chart shows the ideal order in which requirements should be implemented if only customer value and development costs were to be considered. In an ideal case, the requirements at the top of the bar chart would have consisted of requirements from release A. The requirements at the top of the bar chart are estimated as having the highest value and the lowest cost and should therefore be implemented as early as possible. The requirements at the bottom are estimated as having the lowest value and the highest cost and should therefore be implemented in a later release, or perhaps not at all.

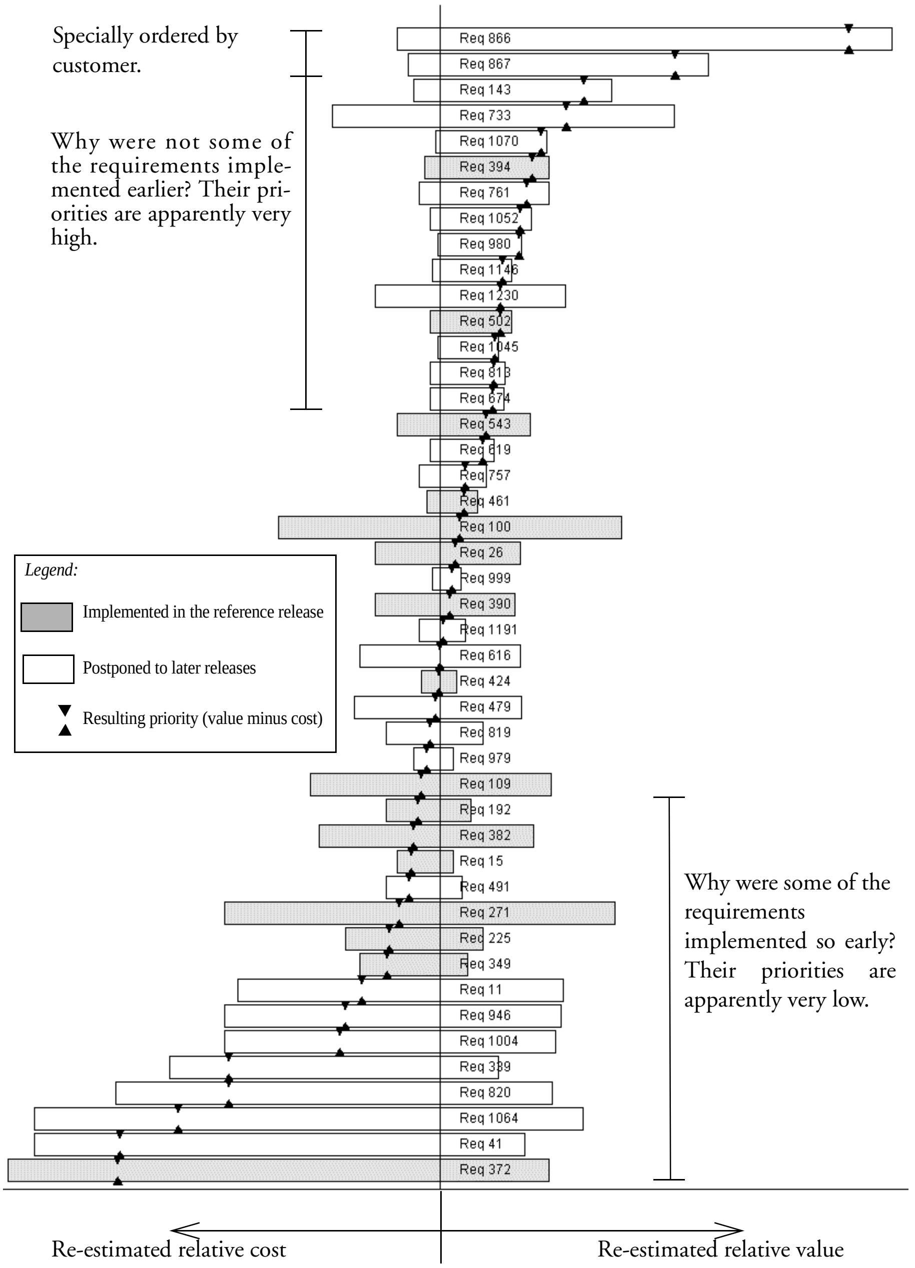
The bar chart illustrates the discrepancies between the two estimation occasions and points out the requirements to discuss. Some of the requirements were not identified in release A, but turned out to be important when they later were identified. Furthermore, requirements interdependencies, release themes and architectural choices complicate the situation and thus this ideal order might not be the most suitable in reality.

Step 3: Root cause analysis.

The bar chart is used in the Root cause analysis, to find out the rationale for the release-planning decisions. The discussion was attended by three representatives from the organisation: one marketing person and two developers, as well as two researchers.

The top 15 requirements were scanned to find the ones that were estimated differently in the re-estimation, i.e. the ones that originate from release B or C. These were discussed to answer the main question "Why was this requirement not implemented earlier?" and motivations to the decision was stated by the participants. In a similar manner, the 15 requirements at the bottom of the bar chart were investigated, to find the ones that originate from release A and B. These requirements were discussed concerning the question "Why was this requirement implemented so early?". Notes were taken of the stated answers for later categorisation of the release-planning decision root causes.

After the meeting, the researchers classified the stated decision root-causes into a total of 19 different categories, inspired by the notes from the meeting. A sheet with the requirements that had been discussed



Specially ordered by customer.

Why were not some of the requirements implemented earlier? Their priorities are apparently very high.

Why were some of the requirements implemented so early? Their priorities are apparently very low.

Figure 4. *Bar chart from the post-release analysis of the requirements in the database using the Focal Point tool.*

during the root cause analysis was compiled, which the organisation representatives used to map the requirements to the found root causes. The result from the classification is displayed in Table 2 and Table 3. Four of the categories were removed as they were not used, leaving 15 root causes.

Step 4: Elicitation of improvements.

The main purpose of the case study was to capture improvement proposals by encouraging the participants to, in connection with each requirement, state some weak areas in need of improvement. This appeared to be difficult since each decision was dependent on the specific context or situation. Therefore, no list of improvement proposals was compiled at this stage. Instead, more generic improvement proposal areas were elicited by investigating Table 2 and Table 3 and the notes taken from the root cause analysis discussion. This is described below.

5.1.3 Results

The case study showed that it was possible to use the PARSEQ method in practice. The release-planning decisions that were made in prior releases could be categorised and analysed and process improvement areas could be identified. The results indicate that the organisation has gained a lot of knowledge since the planning of the reference release, which is a promising sign of evolution and progress.

The causes for implementing requirements earlier than necessary are shown in Table 2. Most of the root causes originate from wishing to satisfy customer demands, either one specific customer or the whole market. However, the evaluation showed that the customer value was not as high as expected. On the other hand, it is difficult to measure "good-will" in terms of money, and therefore these decisions may not be essentially wrong. Other root causes for implementing requirements earlier than necessary concern implementation issues, such as incorrect effort estimations, which lead us to believe that estimations ought to be more firmly grounded. Another reason concerns release themes which is a kind of requirements interdependency that is necessary to take into account. Developing and releasing small increments of requirements, in order for customers to give feedback early, is a good way of finding out more exactly what customers want, while assigning a low development effort.

Table 2. "Why was this requirement implemented so early?"

	Root Causes	Req 192	Req 382	Req 15	Req 271	Req 225	Req 349	Req 372	Req 41
Implem. issues	RC1: Under-estimation of development effort	■	■		■				
	RC2: Part of release theme		■		■			■	■
	RC3: A quick fix to provide customers opportunity to give feedback		■		■				■
Customer issues	RC4: Requirement ordered by a specific customer								■
	RC5: Requirement specifically important for a key customer				■			■	■
	RC6: Over-estimation of customer value			■		■	■	■	■
	RC7: Impressive on a demo					■		■	■
	RC8: Competitors have it, therefore we must also have it							■	
	RC9: Competitors do not have it; gives competitive advantage			■		■			■

Table 3. "Why was this requirement not implemented earlier?"

	Root Causes	Req 143	Req 733	Req 1070	Req 761	Req 1052	Req 980	Req 1146	Req 1045	Req 813	Req 674	Req 866	Req 867
Implementation issues	RC10: Over-estimation of development effort	■							■	■			
	RC11: Insufficient understanding of scale-up effects			■				■					
	RC12: No good design solution available		■		■								
	RC13: Sub-optimal decision based on requirements partitioning		■										
	RC14: Only partial implementation in a first increment				■	■	■				■		
Cus. issues	RC15: Requirement ordered by a specific customer											■	■

As Table 3 shows, the reasons for implementing requirements later than optimal mainly apply to implementation issues. The category complying with the most requirements regards partial implementation in a first increment, which means that it was implemented earlier, but only partially and therefore the requirement remains.

The root cause tables and the material from the discussion were used in the investigation of possible improvement areas. Five areas were found, which could be linked to the root causes. The improvement areas are described below.

Trim the division of large requirements into smaller increments.
The manner in which large requirements, affecting several components or having a large implementation effort, are divided into smaller increments can be more thoroughly investigated. The division can be done for several reasons: to get customer feedback at an early stage, to investigate alternative design solutions or to make small incremental improvements of the functionality. Root causes number 3 and 14 deal with requirements developed in increments and the discussions resulted in the idea that the organisation would benefit from an improved increment planning.

Enhance the overall picture of related requirements.
Some requirements were acknowledged as being related to other requirements due to involving the same feature. These would probably have benefited from creating an overall picture of the release so that all aspects of the specific feature were accounted for. In some cases a feature involved several requirements and after implementing some of them the developers felt content. The related requirements could instead have been designed concurrently in one larger action to avoid sub-optimal solutions. It would also have helped in identifying the most important requirements for that feature. These requirements relations could be taken into consideration more carefully as root cause number 13 describes.

Additional elicitation effort for usability requirements.
It was recognised that the requirements dealing with the user interface did not fulfil some special customer needs, as described by root cause number 11. The problem concerned scale-up effects and could have been discovered through a more thorough requirements elicitation. Actions to take include building prototypes and consulting customers with special user interface needs.

Improve estimations of market-value of features in competing products.
It seems that many requirements were implemented with the objective of outperforming competitors, as reflected in root cause number 7, 8 and 9. However, looking too much at what competitors have or what may look nice on a prototype or demo may bring less value to the product than expected. The value estimations of the competitors' products may need to be improved.

Improve estimations of development effort.
Root causes number 1 and 10 concern over- and underestimations of the development effort. Results from an earlier study indicate that the release plan is very dependent on accurate time estimates, since the estimates affect how many of the requirements that are selected (Paper 1). Underestimation may result in an exceeded deadline and over-estimation may exclude valuable requirements. Improving this area may enhance release-planning and requirements selection quality.

The case study participants found the one-day exercise interesting and instructive. They all agreed that it was valuable to reassess previous releases and reflect on the decisions made. It was during the root cause analysis the most learning occurred since the discussions between the participants were very fruitful. A set of improvement issues to bear in mind during requirements selection was assessed as valuable for future releases.

Despite the fact that 20 out of 45 requirements were assessed as belonging to the wrong release, only a few of the 20 requirements decisions were essentially wrong. Keeping in mind the knowledge available at the time of the reference release, most release-planning decisions were correct, i.e. market opportunities and risks have to be taken, incremental development is applied and only a limited amount of time can be assigned to requirements elicitation and evaluation.

5.1.4 Threats to Validity

There are a number of validity issues to consider in the case study. First of all, data were not extracted from a representative sample as 15 requirements were selected from each release although the releases varied in size. Therefore it might not be possible to generalise the improvements found to the whole requirements repository. Since the data only included requirements that were implemented, and no rejected requirements, there would be more decisions to consider in a more thorough evaluation. Furthermore, a different set of requirements would possibly generate a different set of root causes, and therefore these shall not be used by themselves. Other improvements could be found if a larger sample was used, including postponed requirements.

The criterion that was used to capture the true value of the requirements appeared to be somewhat difficult to use. Since the development cost was known in most cases, it was difficult for the participants to concentrate on the customer value only, without implicitly taking the cost into account. It was also difficult to, in retrospect, consider the reference release and the value at that particular time without regard of the situation today. This might have had a slight effect on the resulting priority of requirements and in the future the criteria could be simplified to improve concentration.

The prioritisation itself is also a source of uncertainty; when not performed thoroughly, the bar chart may not show the appropriate requirements priorities. Since the prioritisation is based on subjective assessments, it is highly dependent on the persons involved. Nevertheless, the consistency check proved that the prioritisation was performed carefully and few judgment errors were made.

Finally, the presented improvement areas are specific to the particular case study organisation and need to be examined in further detail to point out the exact measures to take. However, the participants state that the exercise itself, imposing thought and reflection, may be more fruitful than the particular improvement proposals.

5.2 Case Study B: In-house Project for Software Product

The PARSEQ method was tried out in a second case study where other techniques for sampling and re-estimation were used. Random sampling from the whole requirements repository was used as we did not select a certain set of releases for the investigation, but instead had the whole system in mind. The re-estimation technique used in this study was more agile than in the prior study, and could be used manually. The intention was to try the PARSEQ method out without commercial tool-support, and therefore a simpler sampling technique was used as well as a manual re-estimation technique.

Below, the case study background is described, along with the operation at the case study site. Finally, the results from the case study are presented and analysed.

5.2.1 Background

The case study organisation develops embedded software products for a global market. The case in focus is an in-house project aimed at improving the IT support for the production system, and its connection to the business system and production database. The production system is divided in three main steps. First, items are provided with software and are tested, then items are labelled and placed in a suitable box, and finally items are bundled into multipacks that should be shipped to a certain retailer. The main project goal was to improve production efficiency, flexibility, maintainability and product traceability.

The project decided to use an incremental development method, with frequent releases, inspired by Extreme Programming (XP) (Beck, 2005). The company had never tried working in an agile manner before, but some co-workers had theoretical experience in agile methodologies.

A repository of requirements was developed early on in the project by interviewing a large number of stakeholder representatives and documenting their wishes for the new system. One of the researchers was involved as assistant in the elicitation of requirements for the system. The project was divided into several releases of the product, and each release was divided into several shorter iterations. In the beginning of each iteration, a Planning game activity was performed in order to decide the iteration content. The requirements repository was used as input to the planning and prioritisation, although the requirements had to be broken down to a more detailed level when planning the iterations.

5.2.2 Conduct of Study

The case study was divided into two separate occasions since the users were unable to attend the first session. The Project manager and the System architect participated during the first session, as they were involved in decision-making throughout the development of the system. The second session was attended by the Production manager and the Production test manager who are key users of the production system.

They also acted as customers of the project and suggested many of the requirements in the repository.

One of the researchers acted as facilitator and took notes during both occasions. The Requirements sampling step took about one hour and was performed by the facilitator before the first session. The sessions on site lasted for approximately two hours each.

Step 1: Requirements sampling.

The requirements repository originally consisted of about 140 requirements. Approximately 20 requirements were removed due to being very similar to other requirements, having evolved out of scope of the project, or being on a too high level.

In order to get a reasonable sample, every fourth requirement was selected, i.e. the sample consisted of 30 requirements. As the system consisted of three main releases that had been launched so far, it was decided to take the whole requirements repository into consideration. Thus the sample included requirements suggested all through the elicitation phase. The sample included both requirements that were implemented in one of the three main releases in the project and requirements that were postponed or excluded.

We intended to use the Planning game (Beck, 2005; Paper 2) as re-estimation technique, since it is possible to perform manually and the participants already had experience in using it. Therefore, each requirement was printed on two separate cards, one for the user value and one for the implementation cost.

Step 2: Re-estimation of value and cost.

The requirements sample was re-estimated based on cost and value. The intention was to let the Project manager and System architect estimate the implementation cost as they represent the developers of the system, and the Production managers estimate the user value as they represent the users of the system. However, the Production managers were unable to attend the first session so the developers had to play both roles.

In this study, the Planning game technique was used for prioritisation of both cost and value. We used the technique to divide the requirements into three groups corresponding to (1) Requirements that are absolutely essential, (2) Requirements that provide added value, and (3) Requirements that are nice to have, for the Value criterion. Similarly the requirements were grouped according to having (1) Essentially higher cost than medium, (2) Medium cost, (3) Essentially lower cost than medium, for the Cost criterion.

Table 4. Number of requirements in each priority group.

Value groups	Developers	Users	Cost groups	Developers	Users
Absolutely essential	16	15	**High**	9	9
Provide added value	6	8	**Medium**	11	10
Nice to have	7	7	**Low**	9	11

The participants were instructed to start with the value criterion, representing the value to the users. They were asked to create three rather evenly large groups, i.e. none of the groups should be almost empty. The three groups were divided as shown in Table 4. The developers only prioritised 29 of 30 requirements, since they classified one requirement as out of scope.

The participants were then asked to rank the cards within each group, in order to get a list of prioritised cards on an ordinal scale. Without instructions, they used a sorting technique based on comparing each card to the others in the list to see where to insert it. They spent a bit more time prioritising the cost criterion than the value criterion, see Table 5. This may indicate that the cost was more difficult for them to estimate than the user value.

In the second session, the users prioritised the requirements based on both value and cost for the purpose of comparing their priorities to the developers'. The users found it easier than expected to prioritise based on value but as can be suspected, the implementation cost was more difficult for the users to estimate. They spent less time on the cost criterion than on the value criterion, probably because the cost was estimated by guessing. The users applied another sorting technique when the three groups had been formed, and divided each group into three new groups until they had reached a complete ranking.

Table 5. Time spent on prioritisation.

Time (minutes)		Developers	Users
Value	Divide into groups	15	13
	Within groups	15	11
Cost	Divide into groups	20	7
	Within groups	20	9

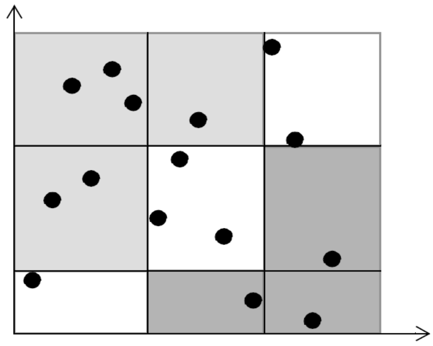


Figure 5. *Example of cost-value diagram based on ranks*

As the users worked rather fast, they were given a chance to use tail-head comparison (Karlsson et al., 1998). The lowest ranked requirement in the top value group was exchanged for the highest ranked requirement in the medium value group before they were satisfied. During prioritisation of implementation cost, they did not change any of the requirements in the tail-head comparison.

The result from the Planning game was illustrated in a cost-value diagram. Figure 5 shows an example of a cost-value diagram drawn based on ranks. The requirements in the upper left field are the ones that should be implemented first as they have a combination of high value and low cost. The requirements in the lower right field are the ones that should be implemented last or not at all, as they have a combination of low value and high cost. The cost-value diagrams were used in the next step, the root cause analysis.

Analysis of agreement. The Kappa value (K) can be used to assess the agreement between a set of raters who assign a set of objects into a set of categories (Siegel and Castellan, 1988). As all participants had to play both the developer and the user role, it was possible to investigate how well the developers' estimates of value correlated with the users' estimates of value, and similarly how well the users' estimates of cost correlated with the developers' estimates of cost.

This was made by comparing the cost-value diagrams created in the two sessions. The three white and grey fields in Figure 5 were used as categories into which objects, i.e. requirements, were assigned. The two sets of raters were the users and the developers. The Kappa values are presented in Table 6. Different suggestions have been presented regarding the interpretation of the Kappa value. In (El Emam, 1999), one

Table 6. Kappa value for a comparison between users' and developers' estimate of cost and value

Criterion	Kappa value
Cost	-0.08
Value	0.32

suggestion is that 0.21<K<0.40 can be regarded as a fair agreement, which would be the case for the agreement between the users' and the developers' estimates of value. Kappa values close to, or below, zero suggest no agreement, and therefore there would be no agreement between the users' and developers' estimates of cost.

Cost-value approach. As the cost estimates made by the users were too unreliable, we have chosen not to include their cost-value diagram here. The value estimates made by the developers turned out to be rather similar to the value estimates made by the users, as the Kappa value indicates. Figure 6 shows the cost-value diagram made from the developers' estimates.

The facilitator drew a diagram on the whiteboard, with the y-axis representing the ranked order of user value and the x-axis representing the ranked order of implementation cost. Then the value cards were used to place the requirements in the correct order based on value, i.e. vertically on the cost-value diagram. The cost cards were used to place them in the correct order based on implementation cost, i.e. horizontally. The facilitator also drew lines representing the three groups, resulting in nine sections (A-I) on the whiteboard, as shown in Figure 6.

Step 3: Root cause analysis.

The root cause analysis consisted of a discussion about the different fields in the cost-value diagram. The project included three main releases, which consisted of several iterations. We agreed that if release planning decision-making were perfect, sections A, B and D would contain requirements from the first release and sections F, H and I requirements implemented in the third release or not at all.

In the cost-value diagram made by the developers, eleven of the twelve requirements that appeared in sections A, B or D were implemented in the system, although five were not implemented in the first release. The discussion showed that those requirements could not be implemented in an earlier release since other requirements were needed as foundation (RC1). They were not needed urgently, but they were very important to

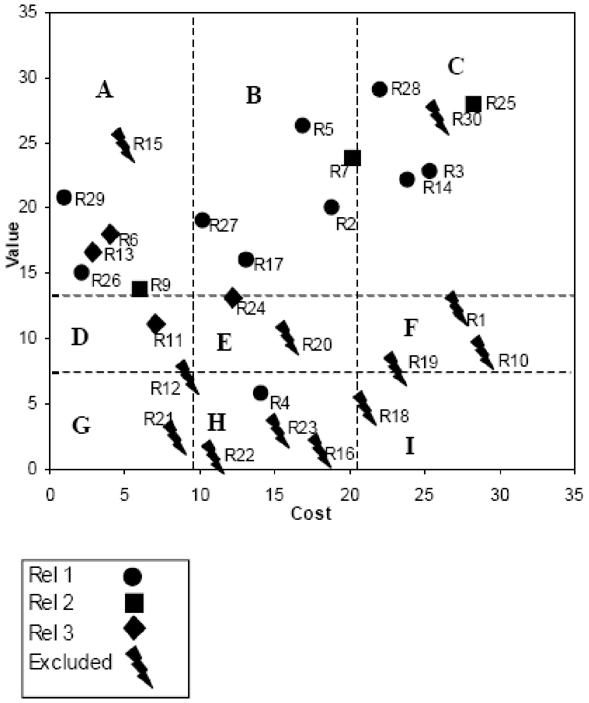


Figure 6. *Cost-value diagram from session 1, with nine sections (A-I) representing the Planning game groups with requirements from different releases.*

include sometime. Therefore they could be postponed to a later release when it was better suited to implement them. Only one of the twelve requirements was excluded. However, the developers stated that the basic problem behind the requirement was solved in another way (RC2).

Sections F, H and I contained eight requirements of which seven had not been implemented. One of the requirements had been implemented in the first release. It was a usability requirement regarding the user interface. It was expected to have a higher user value during development than it actually had when the system was put into operation (RC3). Therefore, it was implemented although it should probably not have been included in the system, with regard to today's knowledge. Table 7 maps the root causes to the requirements.

It was difficult to draw any conclusions from the root cause analysis performed in the second session, i.e. with the users. The cost-value diagram pointed out some requirements to investigate, but since there were no developers to answer questions about if and why requirements were implemented in a certain release, it was difficult to find any root

Table 7. Root causes for the in-house project.

	R4	R6	R7	R9	R11	R13	R15
RC1: Foundation in the architecture did not exist yet		■	■	■	■	■	
RC2: Other functionality solved the problem							■
RC3: Over-estimation of user value	■						

causes. It was also difficult to trust the indications from the cost-value diagram, since the users knew their cost estimates were unreliable. Therefore it was decided to only use the users' estimates as comparison with the developers' estimates, which was presented earlier.

Step 4: Elicitation of improvements.

The PARSEQ analysis indicated that the release planning decision-making in this project was successful. The developers concluded that one reason for the successful release planning was the iterative development. During the project, the Planning game was used to evaluate and prioritise the requirements regularly, and the release plan was flexible enough to adapt to changes in the requirement priorities and in the project resources.

Thus, the most important insight was that regular prioritisation yields better release plans and this lesson learned will be brought into other projects at the company. More prototyping activities during release planning was also mentioned as a possible improvement in other projects. The participants were pleased with the result, as it was a confirmation that they had prioritised and decided correctly and that iterative development is a successful way of working.

Although the users' root cause analysis did not bring any particular conclusions, the users were surprised with the extent to which they were actually given the most important functionality. Since the scope of the project was cut down during development due to lack of resources, the users were worried that some important functionality had been excluded. The retrospective analysis showed that this was not the case, which of course is encouraging for all stakeholders.

5.2.3 Results

The results from the PARSEQ analysis in this case study indicate that the release planning decision-making in the investigated project has been successful. Two main reasons for the successful release planning have been discussed. First of all, as suggested by the developers, the iterative development and continual re-prioritisation provided a flexible release plan that could be revised and adapted to changes in the requirement priorities and in the project resources. Prototyping activities were also mentioned to have improved release planning as user feedback could be taken into consideration.

A second possible reason is the type of project that was investigated. In the previous case study, a market-driven product was investigated in the retrospective, while in this case study an in-house project was put under investigation. The users in an in-house project are few and more tangible, and have more similar requirements for the system. Users of a commercial product can use the system in many different ways, sometimes in ways unknown to the developer. The users' requirements are therefore more scattered and diverse for a market-driven product. This may be one of the reasons for release planning appearing successful - the users' needs were easier to find early on and the developers understood the users' opinions as they are all within the same company.

One goal of this case study was to investigate a more agile approach to the PARSEQ method, in comparison to the previous case study. Since most projects do not have requirements stored in a commercial tool, it is interesting to investigate how a manual prioritisation technique works for the second step of the PARSEQ method. According to the participants it was easy to use the Planning game procedure to rank the requirements. However, the Planning game only presents the requirements on an ordered list, while the RM tool, used in the previous case study, also presents the ratio between requirements priorities. This difference may affect the results so that another set of requirements is pointed out in the cost-value diagram. This is because in this case the cost-value diagram is based on ranks, while in the previous case it is based on ratios, so some requirements might end up in another root cause area. The ratio scale provides more information that can be valuable for decision-making. However, it seems as the Planning game may be sufficient for our purpose.

5.2.4 Threats to Validity

Ideally, the cost-value diagram should be designed using developers' cost estimates and users' value estimates and the retrospective session should be attended by representatives from both roles. As the analysis was divided in two sessions in the case study, it may have affected the results. The developers' estimates of value agreed rather well with the users' estimates, but if the users' estimates had been available, a slightly different set of requirements would possibly have been pointed out in the root cause analysis. As this kind of analysis depend on subjective views, it is important to select the right participants. In the future, both users and developers should be present to enforce an interesting discussion between the parties. The threat was reduced by letting two user representatives and two developer representatives co-operate and negotiate on the priorities.

The sample was selected from a list of requirements arranged by date of arrival. Therefore, no consideration was made regarding which release the requirements belong to. Only a few were implemented in the second and third releases, while many were implemented in the first release. Requirements more evenly distributed between the releases can improve the possibility to generalise the results to the whole requirements repository. Similarly, the analysis could have been improved if less requirements were postponed.

The requirements repository that was used as input to the study contained high-level user requirements and therefore it was sometimes difficult for the participants to judge whether or not a requirement had been implemented - some were implemented partially and some were implemented over a long period of time. Therefore, the mapping between requirements and release number is approximated. It would possibly have been easier to do the mapping if more detailed requirements were used instead of user requirements. However, it is desired to get the users' view of the system, which may be difficult if analysing too detailed requirements.

Some issues regard the method rather than the particular case, and need further attention in future case studies. Several of the requirements in the analysis were subject to interdependencies. Some requirements affected the system architecture and had to be implemented before others. This complicated the re-evaluation. The developers tended to give the more fundamental requirements a higher priority, as they needed to be implemented early on. Thus, the value to the users had to stand back.

This may be one explanation to the difference between users' and developers' estimates of value.

5.3 Case Study C: Product Line Development for an Embedded Product

The third case study was performed at a company developing embedded software products, using a product line approach. The first in a series of products in a product line was selected as reference release (called release A). 42 high-level requirements that had been candidates for the product were selected for analysis in a series of workshops. The PARSEQ prototype tool was used to support the method. The analysis resulted in a set of root causes and improvement proposals.

5.3.1 Background

The case study organisation develops embedded software products sold on an open consumer market. New products in the product family are released several times a year, all based on the same product platform, see Figure 7. A release is called a *heartbeat* because of its regular frequency. Different products may target different market segments and types of users. Therefore, the products are different from the users' point of view although they belong to the same product family. For each product, one Maintenance Release (MR) is released at the same time as a new product is released based on the same platform. The maintenance releases may include additional requirements but are mainly used for error correction.

Requirements are elicited from several different customers, user groups, and sub-contractors. Decisions regarding which features to

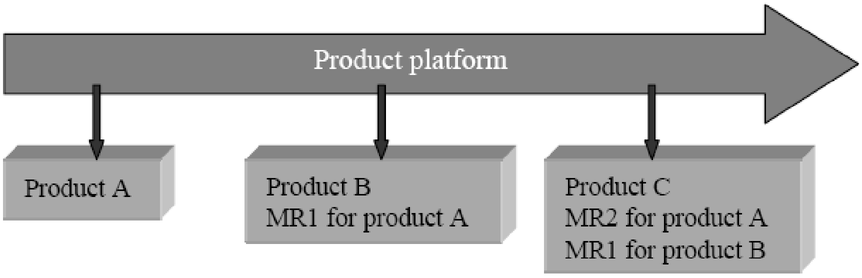


Figure 7. *Relation between product platform and individual products. One or more Maintenance Releases (MR) are released at the same time as a new product release.*

include in the products are made in several steps. In the investigated case, the product scope was set too large in the beginning so that it had to be de-scoped later on in the development process. The de-scoping was performed by product planners, product managers and project managers a number of times during development in order to reduce and change the product scope. The requirements subject to de-scoping represent difficult release planning decisions especially when resources are scarce and need to be used for the most important requirements.

The release planning in this case differs from the one investigated in earlier case studies. A new "release" in the product line case is in fact a new product, and several products in the product line can be available at the market place in parallel. In earlier cases we have investigated release planning for one product which is updated in several consecutive releases. Then earlier releases are replaced and need no further attention.

5.3.2 Conduct of Study

The case study was divided into a set of meetings and workshops and was also followed up by e-mailing and telephone calls. First, two information meetings were held in order to explain to relevant personnel the purpose of the PARSEQ analysis and to identify appropriate personnel for the following workshops. At the first meeting, the requirements engineer for the product in focus was chosen to conduct the requirements sampling so that the first step of PARSEQ could be started. At the second meeting the first version of the requirements sample was discussed and some suggestions for additional requirements were posed. The cost- and value criteria were also discussed. The participants agreed that the cost should be defined as development effort, and the value as business value to the company. In addition, two persons were selected for the re-estimation step: a product manager for re-estimating the business value of the requirements, and a technical project manager for re-estimating the development effort. Thereafter, the re-estimation step was conducted in two separate sessions and the root cause analysis and elicitation of improvement steps were conducted in one workshop. The first author acted as facilitator at all workshops.

Step 1: Requirements sampling.

The sampling step was conducted by the requirements engineer for the product in focus. The product had been in the markets for about 8 months. Since then, another product based on the same platform had been released and a third one was planned for release in the near future.

Notes from the de-scoping meetings held during development were used as basis when hand picking the requirements for the sample. Thereby, the selected requirements had all been candidates for the product and each requirement had been subject to the difficult decision to remove it or include it in the product. Therefore, the sample was interesting from the point of view of release planning decision-making.

In total, 42 requirements were selected for the sample, among which 25 were implemented in the reference product A, 9 were implemented in the consecutive product B, and 8 were excluded, but might be implemented in future products. The requirements were all high-level features, described as one-liners.

Step 2: Re-estimation of cost and value.

The re-estimation of cost and value was performed during two separate sessions. During the first one, the product manager estimated the business value for the product requirements, and during the second one two technical project managers estimated the development effort. The PARSEQ tool was used on both occasions and the prioritisation technique Planning game was used for the re-estimations.

Re-estimation of business value: The product manager had prepared for the exercise by dividing most of the requirements into three groups regarding their business value. The criterion business value was defined as a mix of strategic, internal, and external value from the company's point of view. The Planning game involves dividing the requirements into three groups: High, Medium, and Low. The product manager defined the High group to include requirements that are essential or "must-haves", i.e., without which the product would not have a clear focus and be possible to sell. The requirements in the Medium group support the product proposition and definition but are not as sensitive to the product topic and not as visible to the customers. The requirements in the Low group are features that do not provide much added value but are nice to have.

The product manager found the task of arranging the requirements into the three groups rather easy. However, when ranking requirements within the groups, several adjustments were done, e.g. one requirement was moved from High to Medium, two requirements were moved from Medium to High, and one requirement was moved from Medium to Low. In addition, when performing a tail-head comparison (Karlsson et al., 1998) between the lowest ranked High requirements and the highest

Table 8. Division into groups from re-estimation of Business value

Business value groups	**Nbr of requirements**
High	17
Medium	12
Low	13

ranked Medium requirements, four requirements were moved from High to Medium, and another five were moved from Medium to High. No changes were made during tail-head comparison between the Medium and Low groups. The High, Medium, and Low group division is shown in Table 8. In addition, when presented with the complete ranking list, the product manager changed the order within the High and Low groups between a handful of requirements. The product manager also stated that it was much more difficult to rank the requirements than to put them into different groups.

Re-estimation of development effort: The project managers estimated the development effort for the requirements. The effort criterion defined by the project managers included both analysis and implementation effort, measured in person-days. Before the meeting they had prepared by looking up the development effort for 11 of the 42 requirements in the sample. It was difficult to find exact effort data for several of the requirements since they involve several different development units. However, since we only aimed at ranking the requirements, the effort data were mainly used for understanding the approximate effort in the High-, Medium-, and Low groups. The development effort of the requirements in the sample varied from 5 to 400 person-days, as can be seen in Table 9.

The project managers found it rather easy to group and rank the requirements by development effort. Table 10 presents the number of requirements in each Planning game group. The effort estimates are somewhat more objective than the value estimates since it is possible to

Table 9. Approximate development effort in the three Planning game groups

Development effort groups	**Development effort**
High	50-400 person-days
Medium	20-50 person-days
Low	5-20 person-days

Table 10. Division into groups from re-estimation of Development effort

Development effort groups	Nbr of requirements
High	10
Medium	14
Low	18

estimate effort in person-days. This may be one of the reasons for the project managers to require less time for the estimation exercise, see presentation of time-consumption in Table 11. The project managers were also more certain about their estimates as they did not change their minds during the exercise, not even when given the possibility to perform a tail-head comparison.

The requirements ranked by business value and development effort were used to create a cost-value diagram (Karlsson and Ryan, 1997) where the x-axis represents the requirements ranked by development effort and the y-axis represent the requirements ranked by business value.

Step 3 and 4: Root cause analysis and Elicitation of improvements. The third and fourth steps, root cause analysis and elicitation of improvements, were conducted in one single workshop. Five people participated: two researchers, one department manager, the product manager, who had estimated business value, and one of the technical project managers, who had estimated development effort. The second project manager was unable to attend the meeting. The researchers had prepared the cost-value diagram to be analysed in the root cause analysis, see Figure 8. The researchers had also prepared a root cause matrix with the requirements that correspond to incorrect decisions. The requirements that were de-scoped or implemented in release B among the requirements in the high contributor areas (see upper striped area in Figure 8) were examined. Similarly, the requirements that were implemented in release A or B among the requirements in the low

Table 11. Time consumption for re-estimation

	Product manager	Project managers
Time to put in boxes	30 min	15 min
Time to rank within boxes	25 min	10 min
Total time incl. discussion	90 min	60 min

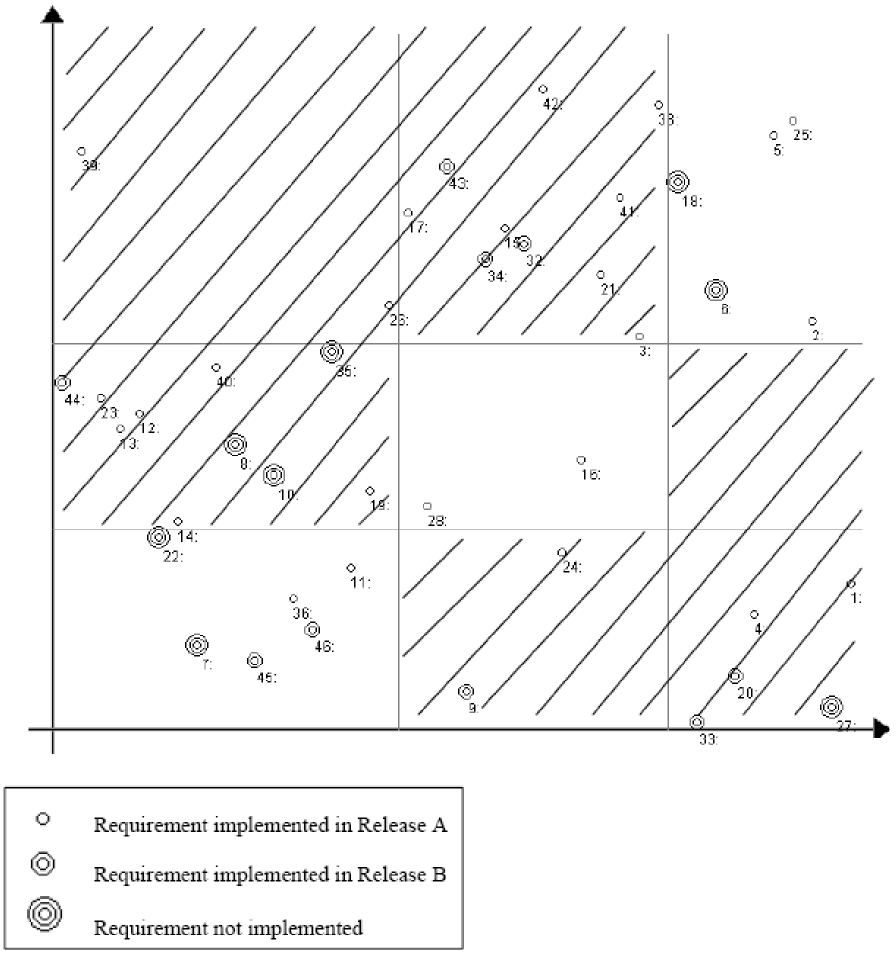


Figure 8. *Cost-value diagram based on ranks*

contributor areas (see lower striped area in Figure 8) were examined. However, due to time limitations we focused mainly on the de-scoped ones in the high contributor areas and the requirements from release A in the low contributor areas since they represent the most interesting decisions-making challenges.

Three requirements (Req Id 1, 4, and 24) in the analysis were de-scoped although they in retrospect were found to be high contributors, and should, therefore, have been included in release A if decision-making were perfect. Similarly, three of the requirements (Req Id 8, 10, and 35) were implemented in release A, although, they were found to be low contributors in retrospect. These six requirements were analysed and the following questions were used to guide the discussion:

- Why was the decision made, when in retrospect it was incorrect?
- When was the decision made? Could it have been discovered earlier?
- How can you prevent similar incorrect decisions in the future?

The questions helped to find root causes for incorrect decisions and improvements for future decision-making.

The facilitator used the root cause matrix to note the root causes and improvement suggestions for each requirement. Table 12 and 13 shows the notes taken during the workshop in root cause matrices for the descoped requirements and for the requirements from release A, respectively.

5.3.3 Results

The result from the workshops show that the PARSEQ method was possible to use for the purpose of finding root causes for incorrect release planning decisions, and improvement suggestions for the process. The main results are the cost-value diagram in Figure 8 and the root cause matrices in Table 12 and 13. The results indicate that although the participants judge the product as a market success, there is room for improvement. Even when analysing successful projects it is possible to find lessons learned based on the participants' experiences. The root causes for making inappropriate release planning decisions can be summarised in five areas based on the root cause matrix:

- Dependencies between requirements
- Lack of resources
- Reliance on customers' judgment
- Dependency on subcontractors
- Architecture and performance enhancements

Some of the root causes are difficult to address. For example, the lack of resources is an issue in almost every development organisation. Especially when several development projects are run in parallel, resources need to be allocated wisely. Similarly, the root cause regarding development of an

Table 12. Root cause matrix for de-scoped requirements

Req Id	Root causes	Elicited improvement proposals
8	-Performance issue, lack of memory. -Dependent on requirement 33 that was de-scoped late. -Resource allocation problem.	-N/A (considered unavoidable)
10	-Dependent on requirement 4, which was included late and was not stable and therefore risky.	-Manage architecture upgrade. -Split milestone in two parts: one for architecture and one for dependent functionality.
35	-Resources cut down as they were needed elsewhere.	-Earlier de-scoping to use the limited resources for the most important functionality. -Increase process flexibility. -Dedicated release themes. -Improve cost-estimation accuracy

Table 13. Root cause matrix for requirements implemented in the reference release

Req Id	Root causes	Elicited improvement proposals
1	-Based on customers' judgment, difficult trade-off between user experience and value proposition. -Business agreement with subcontractor went out, needed in-house solution.	-Consider user opinions, not only customers' opinions. -Focus on the product position. -Early analysis of end user perspective on e.g. user experience, usability and usage frequency.
4	-Needed from architecture design viewpoint, long-term improvement.	-Balance between long term and short term business value. -Increase efficiency and productivity during development.
24	-Technical dependency on requirement 16, which was developed in release A. -Needed improved performance.	-Figure out dependencies from a marketing perspective in de-scoping meetings.

in-house solution instead of the subcontractor one is a necessary action to take in that particular situation. However, it can always be discussed how these issues could be prevented or discovered earlier on in the development.

The improvements that were suggested are summarised and discussed below. To structure the improvements they are assembled in three groups regarding the development process, market issues and development issues.

Improvement of the development process:

- *Split milestone in two parts: one for architecture and one for dependent functionality.* This was suggested by the project manager in order to retain focus on architecture improvements. This would however require major rework of the process and might not be realistic to implement at this point.
- *Earlier de-scoping* was discussed as a means for focusing resources on the most important functionality that will definitely not be de-scoped. The limited resources are needed for e.g. stability issues in the end of the project and cannot be allocated solely to features. Earlier de-scoping would provide better planning, but there is a general resistance to de-scoping as people fear losing valuable requirements.
- *Increase process flexibility* in order to cope with changes in functionality or resources. The development schedules are too tight when unforeseen changes occur. However, due to deadlines and competition, it is difficult to motivate this type of process relaxation.
- *Increase efficiency and productivity during development* so that resources are available for architecture improvements. However, no suggestions were stated regarding how to achieve increased development efficiency.

Improvement of market issues:

- *Consider user opinions, not only customers' opinions.* The difficult trade-off between user experience and customer value proposition was discussed by the product manager. It is difficult to get user opinions at an early stage of development and therefore the organisation relies on the customers' demands and wishes.
- *Focus on the product proposition* and how the product is marketed in order to use the resources in the best way possible. The product proposition should be in focus when allocating the scarce resources.
- *Early analysis of end user perspective, e.g. user experience, usability and usage frequency.* It is, however, difficult to get user opinions at an early stage of development and if it comes too late it is difficult to take into consideration.

- *Figure out dependencies from a marketing perspective already in the de-scoping meetings.* Several requirements dependencies were discovered in the analysis and these would have been valuable to discover during planning so that development resources could be optimised. However, there is a need for techniques to identify dependencies, and it is difficult to find them early on in the process.

Improvement of development issues:

- *Architecture upgrade* is needed to cope with e.g. stability issues in dependent requirements. Difficult trade-offs between architecture improvement and development of functionality that gives visible value to the users was discussed several times.

- *Dedicated release themes* with the same product proposition in all included products would make development more efficient. This is, however, difficult since the market require each heartbeat to have innovation and a differentiated proposition line-up. It requires good planning and timing. It is further complicated by the fact that there may be 2 years between planning and product release, which is a long time since changes in the markets need to be managed.

- *Improve cost-estimation accuracy* so that resource usage is optimised. It may require introducing cost-estimation techniques since it is mainly done in an ad hoc manner today.

- *Balance between long term and short term business value.* Although architecture improvements are not as visible to the users, it is important to include those requirements so that the foundation for features is available when needed.

Other experiences were also discovered during the analysis. When the cost-value diagram was presented, the participants agreed that a similar approach would be useful also in planning and de-scoping discussions in order to consider and visualise the trade-offs between costs and value. One participant stated that if a high value feature is extremely expensive, it should reduce the overall priority. An interesting correlation was discovered during the root cause analysis. The cost-value diagram (see Figure 8) illustrates that most requirements that have a high value also have a high development effort and very few functions are both inexpensive and valuable. Dependencies between requirements were discovered during the workshops. For example, two pairs of requirements were duplicates from the value point of view since they end up in the same feature. However, they had different development effort since one requirement is required for the other one to function. This kind of dependency is called *REQUIRES* (Carlshamre et al., 2001) and regards the implementation order of requirements. It is common in platform development that platform requirements are needed before it is possible to add application requirements.

5.3.4 Threats to Validity

The estimates of effort and value are to some extent subjective. In addition, different persons were involved in planning the release and the retrospective re-estimation of effort and value. However, as described in Section 5.3.2, available effort estimates were used for more than a quarter of the requirements, which improves the validity of the effort ranking. The value estimates are even more subjective than the effort estimates as it is difficult to measure value in numbers. However, a marketing report was used to validate the value ranks. The top five features in the ranking list were also found in the marketing report, judged as "must-haves" by the user representatives. There is not a perfect mapping between the features in the sample and the features in the marketing report, since the features in the marketing report also involve hardware, which is not in focus in this study. Otherwise, we would possibly have found additional concurring features.

For some of the requirements it was difficult to state whether or not it was developed; because of the high abstraction level some requirements were developed partly or developed over a long period of time. However, the requirements engineer involved in sampling used his best estimates. The judgment was confirmed by the project managers during the re-estimation session.

There might be some bias due to the fact that the product manager knows that the product was a market success and knows which requirements that were providing added value and not. Therefore, some of the postponed requirements might get a lower value because the product manager knows that the product managed without them. This bias is difficult to avoid because the participants cannot avoid using the information they have about the release plan.

As can be seen in Table 9, the distribution of effort numbers is not linear. If effort numbers were available for all the requirements, it would have been possible to prioritise using the ratio scale instead of the ordinal scale. Then the cost-value diagram could have looked differently and perhaps another set of requirements would have been selected for analysis in the PARSEQ workshop. However, other research has shown that the cost-value diagram based on the ordinal scale is substantially similar to the one based on the ratio scale, see Part II of the thesis. Therefore, the cost-value diagram based on ranks is assumed to be sufficient as decision-support in the PARSEQ method.

The improvements found in the retrospective analysis are specific to the investigated case. Some improvements are related to the product line approach used in the organisation and could possibly be useful to other projects in the organisation working in the same manner.

6. Discussion

The PARSEQ method is relevant to decision-making on two different levels. First of all, the method aims at finding incorrect release planning decisions so that more successful release planning can be conducted in the future. Secondly, decision-making is involved when deciding on how to change the current way of working, based on the improvement proposals that are found in the analysis. The three case studies presented in Part III are examples of the first kind of decision-making. The second kind of decision-making has not been studied in the presented cases since it requires a more long-term investigation to implement and evaluate the suggested improvements.

In this part of the thesis we have presented three consecutive case studies investigating the PARSEQ method. In the first case study, a commercial RM tool was used for the first steps of PARSEQ. As many organisations manage their requirements using simple spread sheets instead of commercial tools we wanted to investigate how the method would work without tool support. Therefore the second and third case study were designed to investigate the method with a more agile and manual technique during the first steps. For the second case study an agile in-house project was selected at a medium-sized organisation. The third case study investigates a large organisation developing embedded consumer products. The differences and similarities of the results are summarised in Table 14 and discussed below.

Table 14. Comparison between case study results.

	Case A	**Case B**	**Case C**
Number of requirements suspected to be in wrong release	20 of 45=44%	7 of 29= 24%	6 of 42=14%
Number of root causes	15	3	10
Number of improvement proposals	5	Lessons learned from good experiences	12
Subjective views	Satisfied participants	Satisfied participants	Satisfied participants

Note that not all of the requirements pointed out in the root cause analysis are necessarily incorrect. This is discovered in the root cause analysis when discussions are held regarding each possible incorrect decision. Different kinds of interdependencies between requirements, such as logical implementation order and bundling of requirements may affect the release planning decisions in addition to user value and development cost. These interdependencies are found in the retrospective analysis.

The outcome of the research questions stated in the Introduction is:

RQ1. Do the participants find the PARSEQ analysis valuable?

The participants in all case studies were asked to describe their opinions and experiences from attending the PARSEQ workshops. All participants stated that the exercise was valuable and several lessons learned were found. In the first case, the PARSEQ method was used successfully as the findings included several improvements to the release planning process found during the root cause analysis. The company discovered that they needed to enhance both the overall picture of related requirements and the division of large requirements into smaller increments. They also found out that usability requirements need more attention in the elicitation phase. The company also tended to estimate the market-value of features in competing products too high, while effort estimates were found to be both too high and too low. The participants found the exercise interesting and instructive.

In the second case, few specific improvements were found during root cause analysis. However, it was possible to draw conclusions regarding the successful release planning and the positive experiences can be used in other projects at the company. Examples of positive experiences are iterative development and continual re-prioritisation, which provides a flexible release plan that can be revised and adapted to changes in the requirement priorities and in the project resources. Prototyping activities were also mentioned to have improved release planning as user feedback could be taken into consideration. The participants found the exercise valuable since it was a confirmation that their releases and iterations were planned with a high degree of certainty.

In the third case, several root causes and improvement suggestions were found for the participating organisation to consider when planning future releases. For example, improvements of the development process include making earlier de-scoping decisions, and increasing the flexibility of the process. Improvements regarding market issues include focusing more on users' requirements and on the product proposition, analysing usability earlier, and analysing dependencies during de-scoping. Improvements regarding development issues involve upgrading the architecture, creating dedicated release themes, improving cost-estimation accuracy, and improving the balance between long-term and short-term business value of requirements. Other experiences discovered during the analysis include using a cost-value diagram to visualise the trade-off between cost and value during planning and development. Several dependencies between requirements were also discovered during the root cause analysis.

The participants concluded the final session with some comments on the method. The project manager stated that it was worth the effort of re-estimation and root cause analysis when the improvement suggestions were discovered. The product manager were satisfied with the result as it confirmed that release planning worked reasonably well, although some outliers, i.e. incorrect decisions, were found.

RQ2. Do the results of the PARSEQ analysis differ depending on the project type, product type and development approach?

The reported studies have different characteristics. Two different project types, market-driven product development and in-house development, are investigated. The cases also represent both pure software products and embedded products. In addition, different development approaches appear: incremental, agile, and product line development. Among the two project types it seems as the PARSEQ method is more applicable to the market-driven situations, although the in-house project participants also found several lessons learned. In the market-driven cases a larger number of root causes and improvement suggestions were found, see Table 14. The results also differ regarding the type of root causes found in the analysis. In the market-driven cases, many root causes for making incorrect release planning decisions involved customer satisfaction issues such as making a demo impressive, keeping up with the competitors, and relying on customers for feedback. In the in-house case, the root causes for most incorrect release planning decisions involved implementation issues, such as providing the foundation in the architecture before implementing certain features. These differences may depend on the project type, as release planning in the market-driven cases is critical in order to be competitive on the market, while the in-house project can focus on releases that fit the development organisation. In addition, the embedded product development using a product line approach entails challenges due to the reliance on sub-contractors and interdependencies between functionality in the platform and in the different applications. It was also more difficult to analyse the product line development since the different releases are actually different products with different focus. Therefore, it is not only a question of timing, and considering the market and the competitors; it is also a question of product proposition and market segmentation.

The differences in characteristics affect the outcome of the PARSEQ analysis. The characteristics need to be considered when planning the retrospective analysis so that the method can be adapted to the particulars of each case.

RQ3. What are the lessons learned about the adaptations of the PARSEQ method to the different cases?

When investigating cases that have different characteristics, it may be necessary to adapt the PARSEQ method to the characteristics of the particular company and product. The lessons learned regarding the adaptation of the PARSEQ method to the three investigated cases are described below. Other adaptations might be needed for other cases.

Sampling may be performed in different ways depending on the repository size and the age of the requirements. As the product in the first case had been on the market for several years, we selected a reference release that consisted of many requirements and was of special interest to the company. In the second case, the sample was taken from the whole repository as it only consisted of 120 requirements. In the third case, the sampling was focused on requirements that had been subject to de-scoping decisions. Therefore, the requirements were interesting to analyse from a release planning viewpoint.

Our adaptations include taking into consideration the prioritisation and release planning tools that the project is used to. In the first case, an

RM tool was already used to store requirements and plan releases and therefore it could be used during the PARSEQ analysis. In the second case, the Planning game was used for release planning and therefore we chose to use a similar approach to the PARSEQ analysis. Since it turned out useful and efficient, the Planning game was also used in the third case, but then with tool-support. The choice of re-estimation technique affects the root cause analysis since the techniques use different scales. The ratio scale provides information regarding the relative distance in priorities while the ordinal scale only provides ranks. This needs to be considered as some situations might benefit more from the ratio scale techniques for example if actual numbers are available on e.g. implementation cost.

It is also possible to imagine variations of the aspects cost and value in the re-prioritisation. In the first and third case, the value concerned business value to the organisation and issues such as new market possibilities, competitors and present customers were taken into consideration. In the second case, the value aspect was purely the users' opinions on which functionality they requested. Thus, different variations of the aspects can be regarded, as long as this is clarified in the beginning of the re-estimation.

RQ4. Can the PARSEQ prototype tool make the PARSEQ analysis more efficient and illustrative?

The tool was a valuable support for the PARSEQ method. Especially during the re-estimation step, it was helpful to be able to choose prioritisation method and criteria for the imported sample. Manual re-prioritisation would probably have been more time-consuming. It was also helpful to have the cost-value diagram automatically generated based on the effort and value estimates. It increased visualization in the root cause analysis step. However, some drawbacks were found during the analysis. The tool lacks support for distributed work since it is not possible to save the results between the re-estimation sessions. The tool need to be improved to support multi-site collaboration since many companies are distributed geographically. In the case study, the facilitator had to insert the ranks manually when creating the cost-value diagram, which evidently required some extra effort. Another disadvantage was found in the root cause matrix. Since only the requirement number was visible and not the feature itself, using the matrix would have required a lot of switching between windows. Instead a matrix was prepared in MS Excel^{TM} with rows representing the requirements and columns open for root causes and improvement suggestions.

Despite these drawbacks, the tool conformed to its purpose of speeding up the analysis and visualising the results.

7. Conclusions

The presented method for retrospective analysis of requirements selection quality, called PARSEQ, was tested in three case studies where candidate requirements for previous releases were evaluated in retrospect. The case studies demonstrate the feasibility of the method in the contexts of the specific cases and the results from the case studies encourage further studies of the method.

The lessons learned from these case studies include:

- Among the investigated types of development, the PARSEQ method seems applicable in the improvement of industrial processes for market-driven requirements engineering, while it seems less suitable for in-house projects.
- Product line development turned out to be more difficult to analyse than software systems since it is not evident that one requirement can be implemented earlier or later, since the different releases are actually different products with different focus. Therefore, it is not only a question of timing, and considering the market and the competitors; it is also a question of product proposition and market segmentation.
- Both developers and users should attend the same session, in order to bring more interesting conclusions from the root cause analysis. This was the case in the first and third case study, but not the second.
- Several of the requirements pointed out in the root cause analysis may actually be implemented in a correct release. This is due to requirements dependencies that force some requirements to be implemented before others, which was particularly visible in the second case study.

- If the retrospective analysis indicates that the release planning has been successful, it is still possible to learn from the analysis. In the second case, prototyping and iterative development appears to have worked well. These are examples of learning from success as described in Section 2.1 (Nolan, 1999).

- Using pre-defined questions in the root cause analysis helps to keep focus. In the third case study, the questions presented in Section 5.3.2 were followed to a large extent but not completely for all requirements as the time was limited. A list of questions for each requirement would have increased the completeness of the data.

- In the third case study, the effort estimates were validated against actual effort numbers provided by the project managers. Similarly, the features with highest value estimates could be validated against the user opinions in a marketing report. This confirms that it is valuable to combine the PARSEQ method with actual project data collected in e.g. a knowledge management effort.

- It could be valuable to use a cost-value diagram during planning, since it provides a possibility to use the original cost-value diagram as a baseline for comparison during the retrospective analysis.

The following areas are interesting to investigate in future case studies of PARSEQ:

- *Process improvement implementation.* Since the investigated projects can not change the process, the discovered process improvements will have to be implemented later on. Therefore, it is not possible to investigate the process improvement introduction within the time frame for these studies. However, it would be valuable to investigate process improvement implementation based on the PARSEQ method in practice in the future.

- *Prioritisation of improvements.* The fifth and final step of the PARSEQ method, i.e. the prioritisation of process improvement proposals has not yet been applied in a case study. Either the cases did not yield that many improvement suggestions or the limited time in the workshops restricted the activities. In the future it would be valuable to investigate improvement prioritisation.

- *PARSEQ prototype tool improvements.* The prototype tool that was used in the third case study needs improvements. The tool was developed based on the needs of the prior case studies in which the re-estimation step, the root cause analysis and the elicitation of improvements steps were performed at the same session. However, in order to perform the analysis in a distributed manner it is needed to save intermediate results between the sessions. This change, among others, would increase the value of the tool.
- *Selection quality metrics.* Given that the requirements sample is representative to the distribution of correct and incorrect decisions, it may be possible to use PARSEQ to provide numerical estimations of the selection quality in terms of fractions of "good" and "bad" decisions. Numerical estimates of selection quality could be used in an analytical model such as the one in Paper 10.
- *Criteria for release planning.* There may be other criteria than cost and value that determines how the releases should be planned, as many different aspects affect the priority in practice. Criteria such as effort, resources, and logical implementation order could be used instead of cost. Further, importance to key customers, importance to users, product strategy, and company profit could be used instead of value. This could be investigated further as the most appropriate criteria may vary depending on situation, product or company in focus.

References

Al-Rawas, A. and Easterbrook, S. (1996) “Communication Problems in Requirements Engineering: A Field Study”, *Proceedings of the 1st Westminster Conference on Professional Awareness in Software Engineering,* Royal Society, London.

Atlas.ti (1997) http://www.atlasti.de (visited September 2006)

Basili, V., Caldiera, G. and Rombach, H.D. (1994) “The Experience Factory,” *Encyclopedia of Software Engineering*, pp. 469-476, John Wiley & Sons, Inc.

Beck, K. (2005) *Extreme Programming Explained*, Addison-Wesley, Boston, MA.

Berander, P. (2004) “Using Students as Subjects in Requirements Prioritization”, *Proceedings of the 3rd International Symposium of Empirical Software Engineering*, Redondo Beach, CA, USA, pp. 167-176.

Berander, P. and Andrews, A. (2005) “Requirements Prioritization”, in: Aurum, A. and Wohlin, C., (eds.), *Engineering and Managing Software Requirements*, Springer-Verlag, Berlin, Germany.

Birk, A., Dingsoyr, T. and Stålhane, T. (2002) “Postmortem: Never Leave a Project without It”, *IEEE Software*, Vol. 19, pp.43-45.

Breitman, K.K., do Prado Leite, J.C.S. and Finkelstein, A. (1999) “The world’s a stage: a survey on requirements engineering using a real-life case study”, *Journal of the Brazilian Computer Society*, Vol. 6, pp. 13-37.

Carlshamre, P. (2001) *A Usability Perspective on Requirements Engineering - From Methodology to Product Development*, Dissertation No. 726, Department of Computer and Information Science, Linköping Studies in Science and Technology, Linköping University, Sweden.

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B. and Natt och Dag, J. (2001) "An Industrial Survey of Requirements Interdependencies in Software Release Planning", *Proceedings of the 5th International Conference on Requirements Engineering*, Toronto, Canada, pp. 84-91.

Carlshamre, P. (2002) "Release Planning in Market-Driven Software Product Development: Provoking an Understanding", *Requirements Engineering*, Vol. 7, pp. 139-151.

Carmel, E. and Becker, S. (1995) "A Process Model for Packaged Software Development", *IEEE Transactions on Engineering Management*, Vol. 42, pp. 50-60.

Carmone, F.J., Kara, A. and Zanakis, S.H. (1997) "A Monte Carlo Investigation of Incomplete Pairwise Comparison Matrices in AHP", *European Journal of Operational Research*, Vol. 102, pp. 538-553.

Carver, J., Jaccheri, L., Morasca, S. and Shull, F. (2003) "Issues in Using Students in Empirical Studies in Software Engineering Education", *Proceedings of the 9th International Software Metrics Symposium*, Sydney, Australia, pp. 239-249.

Chatzoglou, P.D. (1997) "Factors Affecting Completion of the Requirements Capture Stage of Projects with Different Characteristics", *Information and Software Technology*, Vol. 39, pp. 627-640.

Cleland, D.I. (1995) *Project Management,* McGraw-Hill, New York.

Coffey, A. and Atkinson, P. (1996) *Making Sense of Qualitative Data*, Sage Publications, Thousand Oaks, California, USA.

Collier, B., DeMarco, T. and Feary, P. (1996) "A Defined Process for Project Postmortem Review", *IEEE Software*, Vol. 13, pp. 65-72.

Collins, C.T. and Miller, R.W. (2001) "Adaptation: XP Style", *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering,* Sardinia, Italy, pp. 54-57.

Cooper, R.G. (2001) *Winning at New Products: Accelerating the Process from Idea to Launch*, Cambridge, Perseus.

Curtis, B., Krasner, H. and Iscoe, N. (1998) "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM*, Vol. 31, pp. 1268-1287.

Damian, D.E. and Zowghi, D. (2003) "RE Challenges in Multi-Site Software Development Organisations", *Requirements Engineering*, Vol. 8, pp. 149-160.

Davis, A.M. (2003) "The Art of Requirements Triage", *IEEE Computer*, Vol. 36, pp. 42-49.

Deifel, B. (1999) "A Process Model for Requirements Engineering of CCOTS", *Proceedings of the Workshop on Requirements Engineering Process*, Florence, Italy, pp. 316-320.

Desouza, K.C., Dingsoyr, T. and Awazu, Y. (2005) "Experiences with Conducting Project Postmortems: Reports versus Stories", *Software Process Improvement and Practice,* Vol. 10, pp. 203-215.

Dingsoyr, T. (2005) "Postmortem Reviews: Purpose and Approaches in Software Engineering", *Information and Software Technology*, Vol. 47, pp. 293-303.

Easton, K.L., McComish, J.F. and Greenberg, R. (2000) "Avoiding Common Pitfalls in Qualitative Data Collection and Transcription", *Qualitative Health Research*, Vol. 10, pp. 703-708.

El Emam, K. and Madhavji, N.H. (1995) "A Field Study of Requirements Engineering Practices in Information Systems Development", *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering*, York, UK, pp 68-80.

El Emam, K. (1999) "Benchmarking Kappa: Interrater Agreement in Software Process Assessments", *Empirical Software Engineering*, Vol. 4, pp. 113-133.

Fenton, N.E. and Pfleeger, S.L. (1997) *Software Metrics - A Rigorous and Practical Approach*, PWS Publishing Company, London, UK.

Fowler, M. and Scott, K. (2000) *UML Distilled. A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley, Reading, Massachusetts, USA.

Glaser, B.G. and Strauss, A.L. (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Weidenfeld and Nicolson, London, UK.

Graaf, B., Lormans, M. and Toetenel, H. (2003) "Embedded Software Engineering: The State of the Practice", *IEEE Software*, Vol. 20, pp. 61-69.

Greer, D. and Ruhe, G. (2004) "Software Release Planning: an Evolutionary and Iterative Approach", *Information and Software Technology*, Vol. 46, pp. 243-253.

Hall, T., Beecham, S. and Rainer, A. (2002) "Requirements Problems in Twelve Software Companies: An Empirical Analysis", *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering,* Keele, UK, pp. 1-17.

Harker, P.T. (1987) "Incomplete Pairwise Comparisons in the Analytic Hierarchy Process", *Mathematical Modelling*, Vol. 9, pp. 837-848.

Herbsleb, J., and Goldenson, D. (1996) "A Systematic Survey of CMM Experience and Results", *Proceedings of the 17th International Conference on Software Engineering*, Seattle, WA, USA, pp. 25-30.

Higgins, S.A., de Laat, M., Gieles, P.M.C. and Geurts, E.M. (2003) "Managing Requirements for Medical IT Products", *IEEE Software,* Vol. 20, pp. 26-33.

Hoepfl, M.C. (1997) "Choosing Qualitative Research: A Primer for Technology Education Researchers", *Journal of Technology Education*, Vol. 9.

Hofmann, H.F. and Lehner, F. (2001) "Requirements Engineering as a Success Factor in Software Projects", *IEEE Software,* Vol. 18, pp. 58-66.

Honour, E. (1995) "Principles of Commercial Systems Engineering", *Proceedings from the 5th Annual International Symposium of the National Council on Systems Engineering*, St. Louis, MO, USA.

Höst, M., Regnell, B. and Wohlin, C. (2000) "Using Students as Subjects - a Comparative Study of Students and Professionals in Lead-Time Impact Assessment", *Empirical Software Engineering*, Vol. 5, pp. 201-214.

IEEE Std 610.12-1990 (1990) "IEEE Standard Glossary of Software Engineering Terminology", Institute of Electrical and Electronics Engineers.

IEEE Std. 830-1998 (1998) "IEEE Recommended Practice for Software Requirements Specifications", Institute of Electrical and Electronics Engineers.

ISO 9000, http://www.iso.org/ (visited September 2006)

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*, Addison-Wesley, Reading, Massachusetts, USA.

Kamsties, E., Hörmann, K. and Schlich, M. (1998) "Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions and Technology Transfer", *Proceedings of the Conference on European Industrial Requirements Engineering*, Hammersmith, UK.

Karlsson, J. (1996) "Software Requirements Prioritising", *Proceedings of 2nd International Conference on Requirements Engineering*, Colorado Springs, Colorado, USA, pp. 110-116.

Karlsson, J. and Ryan, K. (1997) "A Cost-Value Approach for Prioritising Requirements", *IEEE Software,* Vol. 14, pp. 67-74.

Karlsson, J., Olsson, S. and Ryan, K. (1997) "Improved Practical Support for Large-Scale Requirements Prioritising", *Requirements Engineering*, Vol. 2, pp. 51-60.

Karlsson, J. (1998) *A Systematic Approach for Prioritizing Software Requirements*, Doctorial Dissertation, Department of Computer and Information Science, Linköping Studies in Science and Technology, Linköping University, Sweden.

Karlsson, J., Wohlin, C. and Regnell, B. (1998) "An Evaluation of Methods for Prioritising Software Requirements", *Information and Software Technology,* Vol. 39, pp. 939-947.

Kauppinen, M., Kujala, S., Aaltio, T. and Lehtola, L. (2002) "Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice", *Proceedings of the 10th International Conference on Requirements Engineering,* Essen, Germany, pp. 43-51.

Kerth, N.L. (2001) *Project Retrospectives: A Handbook for Team Reviews,* Dorset House Publishing, New York.

Landis, J.R. and Koch, G.G. (1977) "The Measurement of Observer Agreement for Categorical Data", *Biometrics*, Vol. 33, pp. 159-174.

Lauesen, S. (2002) *Software Requirements - Styles and Techniques,* Addison-Wesley, Harlow.

Lauesen, S. and Vinter, O. (2001) "Preventing Requirements Defects: An Experiment in Process Improvement", *Requirements Engineering*, Vol. 6, pp. 37-50.

Leffingwell, D. and Widrig, D. (2000) *Managing Software Requirements - A Unified Approach*, Addison-Wesley, Reading, Massachusetts, USA.

Lehtola, L., Kauppinen, M. and Kujala, S. (2004) "Requirements Prioritisation Challenges in Practice", *Proceedings of the 5th International Conference on Product Focused Software Process Improvement,* Kansai Science City, Japan, pp. 497-508.

Lehtola, L. and Kauppinen, M. (2004) "Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects", *Proceedings of European Software Process Improvement Conference*, Trondheim, Norway, pp 161-170.

Lehtola, L. and Kauppinen, M. (2006) "Suitability of Requirements Prioritization Methods for Market-Driven Software Product Development", *Software Process Improvement and Practice,* Vol. 11, pp. 7-19.

Lubars, M., Potts, C. and Richter, C. (1993) "A Review of the State of the Practice in Requirements Modelling", *Proceedings of the 1st International Symposium on Requirements Engineering,* San Diego, California, USA, pp. 2-14.

Miller, E. (1998) "Managing Embedded Software", *Computer-Aided Engineering*, Vol. 17, pp. 58.

Moisiadis, F. (2002) "The Fundamentals of Prioritising Requirements", *Proceedings of the Systems Engineering, Test and Evaluation Conference,* Sydney, Australia, pp. 108-119.

Myers, M.D. (1997) "Qualitative Research in Information Systems", *MIS Quarterly, MISQ Discovery,* Vol. 21, pp. 241-242.

Newkirk, J.W. and Martin R.C. (2001) *Extreme Programming in Practice*, Addison-Wesley, Harlow.

Nikula, U., Sajaniemi, J. and Kälviäinen, H. (2000) "A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises", *TBRC Research Report 1*, Telecom Business Research Center Lappeenranta, Lappeenranta University of Technology, Finland.

Nolan, A.J. (1999) "Learning from Success", *IEEE Software*, Vol. 16, pp. 97-105.

Novorita, R. and Grube, G. (1996) "Benefits of Structured Requirements Methods for Market-Based Enterprises", *Proceedings of the International Council on Systems Engineering (INCOSE) 6th Annual International Symposium on Systems Engineering: Practices and Tools*, Boston, Massachusetts, USA.

Patton, M.Q. (2002) *Qualitative Research and Evaluation Methods*, Sage Publications, California, USA.

Paulk, M.C, Curtis, B., Chrissis, M. and Weber, C. (1993) "Capability Maturity Model for Software (Version 1.1)", *Technical Report CMU/SEI-93-TR-024*, Carnegie Mellon, Software Engineering Institute.

Paulk, M.C., Weber, C.V. and Curtis, B. (1995) *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley, Reading, Massachusetts, USA.

Potts, C. (1995) "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software", *Proceedings of the 1st International Symposium on Requirements Engineering*, York, UK, pp. 128-130.

Regnell, B. Beremark, P. and Eklund, O. (1998) "A Market-Driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme", *Requirements Engineering,* Vol. 3, pp. 121-129.

Regnell, B., Höst, M., Natt och Dag, J., Beremark, P. and Hjelm, T. (2001) "An Industrial Case Study on Distributed Prioritization in Market-Driven Requirements Engineering for Packaged Software", *Requirements Engineering,* Vol. 6, pp. 51-62.

Regnell, B., Karlsson, L. and Höst, M. (2003) "An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development", *Proceedings of the 11th International Requirements Engineering Conference,* Monterey Bay, California, USA, pp. 254-263.

Robson, C. (2002) *Real World Research*, Blackwell, Oxford.

Royce, W.W. (1970) "Managing the Development of Large Software Systems: Concepts and Techniques", *Proceedings of IEEE WESTCON*, pp. 1-9.

Ruhe, G., Eberlein, A. and Pfal, D. (2002) "Quantitative WinWin: a New Method for Decision Support in Requirements Negotiation", *Proceedings of the 4th International Conference on Software Engineering and Knowledge Engineering*, pp 159-166.

Runeson, P. (2003) "Using Students as Experiment Subjects - an Analysis on Graduate and Freshmen Student Data", *Proceedings of the 7th International Conference on Empirical Assessment and Evaluation in Software Engineering*, Keele, UK, pp 95-102.

Rus, I. and Lindvall, M. (2002) "Knowledge Management in Software Engineering", *IEEE Software*, Vol. 19, pp. 26-38.

Saaty, T.L. (1980) *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*, McGraw-Hill, New York.

Sangiovanni-Vincentelli, A. and Martin, G. (2001) "Platform-Based Design and Software Design Methodology for Embedded Systems", *IEEE Design & Test of Computers*, Vol. 18, pp. 23-33.

Sawyer, P., Sommerville, I. and Kotonya, G. (1999) "Improving Market-Driven RE Processes", *Proceedings of the International Conference on Product Focused Software Process Improvement*, Oulu, Finland, pp. 222-236.

Sawyer, P. (2000a) "Packaged Software: Challenges for RE", *Proceedings of the 6th International Workshop on Requirements Engineering: Foundation for Software Quality*, Stockholm, Sweden, pp. 137-142.

Sawyer, P. (2000b) "Packaged software: implications of the differences from custom approaches to software development", *European Journal of Information Systems,* Vol. 9, pp. 47-58.

Schalken, J., Brinkkemper, S. and van Vliet, H. (2006) "A Method to Draw Lessons from Project Postmortem Databases", *Software Process Improvement and Practice,* Vol. 11, pp. 35-46.

Shen, Y., Hoerl, A.E. and McConnell, W. (1992) "An Incomplete Design in the Analytic Hierarchy Process", *Mathematical Computer Modelling,* Vol. 16, pp. 121-129.

Siddiqi, J. and Shekaran, M.C. (1996) "Requirements Engineering: the Emerging Wisdom", *IEEE Software,* Vol. 13, pp. 15-19.

Siegel, S. and Castellan, J.N. (1988) *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York.

Sommerville, I. (2001) *Software Engineering*, Addison-Wesley, Harlow.

The Standish Group International Inc. (2001) "Extreme CHAOS", *http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf* (visited July 2006)

SWEBOK (2004), http://www.swebok.org (visited July 2006)

Telelogic (2006) http://www.telelogic.com/corp/products/focalpoint/index.cfm (visited July 2006)

Tichy, W.F. (2000) "Hints for Reviewing Empirical Work in Software Engineering", *Empirical Software Engineering,* Vol. 5, pp. 309-312.

Ulrich, K.T. and Eppinger, S.D. (2000) *Product Design and Development*, McGraw-Hill, Boston.

Wiegers, K. (1999) *Software Requirements*, Microsoft Press, Redmond, WA.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. and Wesslén, A. (2000) *Experimentation in Software Engineering - An Introduction*, Kluwer Academic Publishers, Boston.

Wohlin, C. and Aurum, A. (2005) "What is Important when Deciding to Include a Software Requirement in a Project or Release?", *Proceedings of the 4th International Symposium on Empirical Software Engineering,* Noosa Heads, Australia, pp. 237-246.

Zhang, Q. and Nishimura, T. (1996) "A Method of Evaluation for Scaling in the Analytic Hierarchy Process", *Proceedings of the International Conference on Systems, Man and Cybernetics*, Beijing, China, pp. 1888-1893.

Reports on Communication Systems

101 **On Overload Control of SPC-systems**
Ulf Körner, Bengt Wallström, and Christian Nyberg, 1989.

102 **Two Short Papers on Overload Control of Switching Nodes**
Christian Nyberg, Ulf Körner, and Bengt Wallström, 1990.

103 **Priorities in Circuit Switched Networks**
Åke Arvidsson, *Ph.D. thesis*, 1990.

104 **Estimations of Software Fault Content for Telecommunication Systems**
Bo Lennselius, *Lic. thesis*, 1990.

105 **Reusability of Software in Telecommunication Systems**
Anders Sixtensson, *Lic. thesis*, 1990.

106 **Software Reliability and Performance Modelling for Telecommunication Systems**
Claes Wohlin, *Ph.D. thesis*, 1991.

107 **Service Protection and Overflow in Circuit Switched Networks**
Lars Reneby, *Ph.D. thesis*, 1991.

108 **Queueing Models of the Window Flow Control Mechanism**
Lars Falk, *Lic. thesis*, 1991.

109 **On Efficiency and Optimality in Overload Control of SPC Systems**
Tobias Rydén, *Lic. thesis,* 1991.

110 **Enhancements of Communication Resources**
Johan M. Karlsson, *Ph.D. thesis*, 1992.

111 **On Overload Control in Telecommunication Systems**
Christian Nyberg, *Ph.D. thesis*, 1992.

112 **Black Box Specification Language for Software Systems**
Henrik Cosmo, *Lic. thesis*, 1994.

113 **Queueing Models of Window Flow Control and DQDB Analysis**
Lars Falk, *Ph.D. thesis*, 1995.

114 **End to End Transport Protocols over ATM**
Thomas Holmström, *Lic. thesis*, 1995.

115 **An Efficient Analysis of Service Interactions in Telecommunications**
Kristoffer Kimbler, *Lic. thesis*, 1995.

116 **Usage Specifications for Certification of Software Reliability**
Per Runeson, *Lic. thesis,* May 1996.

117 **Achieving an Early Software Reliability Estimate**
Anders Wesslén, *Lic. thesis*, May 1996.

118 **On Overload Control in Intelligent Networks**
Maria Kihl, *Lic. thesis,* June 1996.

119 **Overload Control in Distributed-Memory Systems**
Ulf Ahlfors, *Lic. thesis,* June 1996.

120 **Hierarchical Use Case Modelling for Requirements Engineering**
Björn Regnell, *Lic. thesis,* September 1996.

121 **Performance Analysis and Optimization via Simulation**
Anders Svensson, *Ph.D. thesis*, September 1996.

122 **On Network Oriented Overload Control in Intelligent Networks**
Lars Angelin, *Lic. thesis,* October 1996.

123 **Network Oriented Load Control in Intelligent Networks Based on Optimal Decisions**
Stefan Pettersson, *Lic. thesis,* October 1996.

124 **Impact Analysis in Software Process Improvement**
Martin Höst, *Lic. thesis,* December 1996.

125 **Towards Local Certifiability in Software Design**
Peter Molin, *Lic. thesis,* February 1997.

126 **Models for Estimation of Software Faults and Failures in Inspection and Test**
Per Runeson, *Ph.D. thesis*, January 1998.

127 **Reactive Congestion Control in ATM Networks**
Per Johansson, *Lic. thesis,* January 1998.

128 **Switch Performance and Mobility Aspects in ATM Networks**
Daniel Søbirk, *Lic. thesis,* June 1998.

129 **VPC Management in ATM Networks**
Sven-Olof Larsson, *Lic. thesis,* June 1998.

130 **On TCP/IP Traffic Modeling**
Pär Karlsson, *Lic. thesis,* February 1999.

131 **Overload Control Strategies for Distributed Communication Networks**
Maria Kihl, *Ph.D. thesis*, March 1999.

132 **Requirements Engineering with Use Cases – a Basis for Software Development**
Björn Regnell, *Ph.D. thesis*, April 1999.

133 **Utilisation of Historical Data for Controlling and Improving Software Development**
Magnus C. Ohlsson, *Lic. thesis,* May 1999.

134 **Early Evaluation of Software Process Change Proposals**
Martin Höst, *Ph.D. thesis*, June 1999.

135 **Improving Software Quality through Understanding and Early Estimations**
Anders Wesslén, *Ph.D. thesis*, June 1999.

136 **Performance Analysis of Bluetooth**
Niklas Johansson, *Lic. thesis,* March 2000.

137 **Controlling Software Quality through Inspections and Fault Content Estimations**
Thomas Thelin, *Lic. thesis,* May 2000

138 **On Fault Content Estimations Applied to Software Inspections and Testing**
Håkan Petersson, *Lic. thesis,* May 2000.

139 **Modeling and Evaluation of Internet Applications**
Ajit K. Jena, *Lic. thesis,* June 2000.

140 **Dynamic traffic Control in Multiservice Networks – Applications of Decision Models**
Ulf Ahlfors, *Ph.D. thesis*, October 2000.

141 **ATM Networks Performance – Charging and Wireless Protocols**
Torgny Holmberg, *Lic. thesis,* October 2000.

142 **Improving Product Quality through Effective Validation Methods**
Tomas Berling, *Lic. thesis,* December 2000.

143 **Controlling Fault-Prone Components for Software Evalution**
Magnus C. Ohlsson, *Ph.D. thesis*, June 2001.

144 **Performance of Distributed Information Systems**
Niklas Widell, *Lic. thesis,* February 2002.

145 **Quality Improvement in Software Platform Development**
Enrico Johansson, *Lic. thesis,* April 2002.

146 **Elicitation and Management of User Requirements in Market-Driven Software Development**
Johan Natt och Dag, *Lic. thesis,* June 2002.

147 **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis**
Håkan Petersson, *Ph.D. thesis,* September 2002.

148 **Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections**
Thomas Thelin, *Ph.D. thesis,* September 2002.

149 **Software Information Management in Requirements and Test Documentation**
Thomas Olsson, *Lic. thesis,* October 2002.

150 **Increasing Involvement and Acceptance in Software Process Improvement**
Daniel Karlström, *Lic. thesis,* November 2002.

151 **Changes to Processes and Architectures; Suggested, Implemented and Analyzed from a Project Viewpoint**
Josef Nedstam, *Lic. thesis,* November 2002.

152 **Resource Management in Cellular Networks -Handover Prioritization and Load Balancing Procedures**
Roland Zander, *Lic. thesis,* March 2003.

153 **On Optimisation of Fair and Robust Backbone Networks**
Pål Nilsson, *Lic. thesis,* October 2003.

154 **Exploring the Software Verification and Validation Process with Focus on Efficient Fault Detection**
Carina Andersson, *Lic. thesis,* November 2003.

155 **Improving Requirements Selection Quality in Market-Driven Software Development**
Lena Karlsson, *Lic. thesis,* November 2003.

156 **Fair Scheduling and Resource Allocation in Packet Based Radio Access Networks**
Torgny Holmberg, *Ph.D. thesis*, November 2003

157 **Increasing Product Quality by Verification and Validation Improvements in an Industrial Setting**
Tomas Berling, *Ph.D. thesis*, December 2003.

158 **Some Topics in Web Performance Analysis**
Jianhua Cao, *Lic. thesis*, June 2004.

159 **Overload Control and Performance Evaluation in a Parlay/OSA Environment**
Jens K. Andersson, Lic. thesis, August 2004.

160 **Performance Modeling and Control of Web Servers**
Mikael Andersson, *Lic. thesis*, September 2004.

161 **Integrating Management and Engineering Processes in Software Product Development**
Daniel Karlström, *Ph.D. thesis*, December 2004.

162 **Managing Natural Language Requirements in Large-Scale Software Development**
Johan Natt och Dag, *Ph.D. thesis*, February 2005.

163 **Designing Resilient and Fair Multi-layer Telecommunication Networks**
Eligijus Kubilinskas, *Lic. thesis*, February 2005.

164 **Internet Access and Performance in Ad hoc Networks**
Anders Nilsson, *Lic. thesis*, April 2005.

165 **Active Resource Management in Middleware and Self-oriented Architectures**
Niklas Widell, *Ph.D. thesis*, May 2005.

166 **Quality Improvement with Focus on Performance in Software Platform Development**
Enrico Johansson, *Ph.D. thesis*, June 2005.

167 **On Inter-System Handover in a Wireless Hierarchical Structure**
Henrik Persson, *Lic. thesis*, September 2005.

168 **Prioritization Procedures for Resource Management in Cellular Networks**
Roland Zander, *Ph.D. thesis*, November 2005.

169 **Strategies for Management of Architectural Change and Evolution**
Josef Nedstam, *Ph.D. thesis*, December 2005.

170 **Internet Access and QoS in Ad Hoc Networks**
Ali Hamidian, *Lic. thesis*, April 2006.

171 **Managing Software Quality through Empirical Analysis of Fault Detection**
Carina Andersson, *Ph.D. thesis*, May 2006.

172 **Fairness in Communication and Computer Network Design**
Pål Nilsson, *Ph.D. thesis*, September 2006.

173 **Requirements Prioritisation and Retrospective Analysis of Release Planning Process Improvement**
Lena Karlsson, *Ph.D. thesis*, October 2006.